Proyecto Final de Carrera

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería Técnica en Informática de Gestión

AI-Live: Generación Automática de Historias de Personajes

Autor: Daniel Morgado Román

Tutor: Daniel Borrajo Millán

Co-Directora: Susana Fernández Arregui

Septiembre de 2015

Agradecimientos

A mis tutores Daniel Borrajo y Susana Fernández, por la ayuda prestada tanto en clase como en la elaboración de este trabajo, la paciencia infinita que han tenido conmigo y las continuas líneas de mejora que me han indicado para el proyecto.

A los alumnos y profesores que han aportado su conocimiento al proyecto AI-Live, haciendo que sea una herramienta de comprensión y estudio del ser humano y no un simple juego.

A mis compañeros de universidad Oscar, Zazo, Helen y Alberto, con los que pasé grandes momentos y me hicieron crecer como persona.

A mis hermanas y familia política, por apoyarme y animarme para continuar con el proyecto y hacerme ver que nunca hay que abandonar.

A mis padres, ya que sin ellos no habría llegado hasta donde estoy, por sus continuos sacrificios con tal de proporcionarnos una educación adecuada.

A Sonia, por estar siempre ahí y tirar de mí cuando he estado desmotivado, por levantarme cuando me he caído, por creer siempre en lo que he hecho y por conseguir sacarme una sonrisa cada día..

Muchas gracias a todos.

Índice de contenidos

Tabla de contenido

Contenido

Índice de Ilustraciones

Índice de tablas

1. Introducción

A pesar de que las primeras computadoras fueron creadas con objetivos puramente científicos, tales como la realización de gran cantidad de sumas matemáticas, o bélicos, poco tiempo bastó para que se emplearan con fines lúdicos.

Partiendo de juegos tan básicos como el "Tres en raya", la evolución en la industria de los videojuegos ha ido creciendo a la misma velocidad que la de las computadoras ofreciendo al cliente final historias más propias del cine que de los videojuegos en sí mismos. De esta forma, y apoyándose en el potencial de los ordenadores, los productores de videojuegos han sido capaces de ir creando mejores juegos, enriqueciendo los guiones, mejorando el apartado gráfico y dotando de una inteligencia artificial a los personajes propia de un ser humano.

1.1 Ámbito del proyecto

Enfocándonos en la inteligencia artificial de los videojuegos, es imposible encontrar un juego en el mercado, a día de hoy, en donde no se haga uso de la misma para crear esa ilusión de inteligencia en los personajes de la historia. El tamagotchi, fable o los sims son claros ejemplos del uso de la inteligencia artificial en personajes.

El proyecto AI-Live es, sin duda alguna, un esfuerzo para conseguir crear una simulación de mundo virtual en donde haya personajes que puedan realizar una vida de forma autónoma satisfaciendo sus necesidades, con comportamientos propios debido a su personalidad, con habilidades que puedan ir mejorando y con una historia propia que pueda ser usada para interrelacionarse entre sí.

Todo comenzó en 2006, realizando un proyecto de simulación de comportamiento social con una arquitectura cliente-servidor. En el juego, los personajes, a los que llamamos actores, toman decisiones basándose en las necesidades y las posibles acciones disponibles en ese momento. Estos actores están implementados como clientes, cuyo código permite al actor tomar las decisiones a realizar sobre su personaje y enviárselas al servidor para que este último se encargue de actualizar el estado del juego tras las acciones del cliente y asignar el turno al cliente que le toque realizar su movimiento.

Al hablar de clientes, en AI-Live nos encontramos con hasta 4 tipos de clientes:

- CLIPS: A través de un sistema basado en reglas, el cliente decidirá con qué acción quedarse para satisfacer sus necesidades.
- CLIPS_MANUAL: Este cliente es como el anterior con la peculiaridad de que la palabra final sobre la acción a realizar caerá en las manos del usuario final, pudiendo seleccionar lo que desea hacer.
- PRODIGY: A partir de un estado, un conjunto de acciones y una descripción este cliente tratará de conseguir el objetivo gracias a un planificador de tareas.
- GUI: Se encarga de mostrar, gráficamente, los estados del juego.

El servidor, por su parte, cuenta con varios motores para poder nutrir a los clientes de sus emociones, habilidades o historias que van haciendo del proyecto AI-Live una herramienta cada vez más potente sobre el comportamiento social de los actores.

1.2 Objetivo del Proyecto Fin de Carrera

Si queremos conseguir que el proyecto AI-Live siga creciendo y simulando, cada vez más, el comportamiento social de los personajes, es necesario que vayamos dotando al mismo de información propia de cualquier ser humano.

El objetivo de este proyecto de fin de carrera es aportar a los personajes una historia propia con eventos que hayan podido ir teniendo a lo largo de su vida. Estos eventos deben ir en consonancia con su personalidad o habilidad y nos encontraremos con eventos que sean compartidos por varios actores al haber una afinidad entre ellos.

La idea que se ha seguido a la hora de generar cada una de las historias es, partiendo de la fecha de nacimiento de cada personaje, establecer nuevas fechas ascendentes y, por cada fecha, se ha formado un evento acorde a los distintos factores propios del personaje así como de los eventos anteriores ya generados.

En la generación de los eventos se tienen en cuenta detalles como la edad del personaje o la localidad en la que está en ese momento así como relativos al tipo de evento en sí (si es un evento relativo a la personalidad del personaje se comprueban sus valores o si es un evento de habilidad se revisan las habilidades).

A medida que se van creando nuevos eventos, las fechas van aumentando hasta llegar a la actualidad que es cuando se decide finalizar la creación de la historia. No tiene un último evento que finalice la historia puesto que lo que se busca es que el personaje

disponga de multitud de eventos que hayan ido ocurriendo a lo largo de su vida y podemos encontrarnos con un último evento de JUBILACIÓN así como uno de TRABAJO.

Como ejemplo, si estamos generando la historia del personaje MIKE, que nació en Madrid en 1950, y la siguiente fecha que se ha establecido para formar un nuevo evento para este personaje es en 1960, los posibles eventos de historia que estarán disponibles para ser vinculados en esa fecha podrán ser eventos infantiles (ir al zoo, ir de excursión, visitar a los abuelos…) pero no habrá eventos relativos con una persona adulta (trabajar, retirarse, casarse..).

Además, aquellos personajes que ya estén en el universo AI-LIVE, con su historia creada, podrán aumentar sus historias con eventos que compartan con los nuevos personajes que se sumen al juego siempre y cuando compartan algún vínculo entre ellos.

Encontraremos eventos genéricos como puedan ser ir al parque o ir de compras, eventos propios de cualquier persona como el nacimiento o ir al colegio, así como eventos un poco más exclusivos como puedan ser casarse o divorciarse.

Este proyecto se centrará en la parte del servidor, ya que es aquí donde se crean los personajes y donde debe crearse la historia de cada uno de los personajes, para poder disponer de la misma desde el primer momento.

Estas historias darán paso a poder comunicarse entre varios actores o a enseñar habilidades a otros de forma que la interrelación entre ellos estará presente a lo largo de la dinámica del juego.

1.3 Estructura del documento

La memoria del proyecto seguirá la siguiente estructura:

- Apartado 1, Introducción: En este apartado se expone una breve introducción al proyecto, el objetivo y la estructura a seguir.

- Apartado 2, Estado del arte: Se realizará un recorrido sobre la evolución sufrida a lo largo del tiempo en los videojuegos y la Inteligencia Artificial usada en ellos. Se ahondará en los métodos que son usados para simular Inteligencia Artificial y se mostrarán ejemplos.

- Apartado 3, Objetivos del proyecto fin de carrera: Se explicarán las metas a alcanzar para conseguir realizar el proyecto de forma correcta.

- Apartado 4, Memoria del trabajo realizado: Se expondrá y explicará el trabajo realizado en este proyecto. Se partirá de una explicación de cómo está montado el sistema AI-LIVE y se explicarán los modelos de conocimiento presentes en el proyecto. Posteriormente se explicarán las soluciones realizadas para abordar el proyecto.

- Apartado 5, Pruebas y resultados: Se mostrarán y explicarán algunas pruebas realizadas con el fin de confirmar que el proyecto cumple con las premisas con las que se partía. Se muestra gráficamente cómo se comporta una historia después de realizar varias ejecuciones con nuevos actores y se explica el porqué de algunos eventos para determinados actores.

- Apartado 6, Gestión del proyecto: Se mostrará cómo se comportaría el proyecto desde un punto de gestión con la estimación de tiempos empleados y los costes que supone tanto a nivel personal, material o indirecto.

- Apartado 7, Conclusiones: Se comentarán los avances introducidos con este proyecto así como las dificultades encontradas a lo largo de la ejecución del mismo.

- Apartado 8, Líneas futuras: Se indicarán posibles mejoras a añadir en AI-LIVE que se han ido ocurriendo mientras se ejecutaba el proyecto.

- Apartado 9, Bibliografía: Se indicarán todas las fuentes consultadas y estudiadas para la realización del proyecto y su memoria.

- Apartado 10, Anexos: Se indicarán todos los documentos creados o actualizados para la elaboración del proyecto.

2. Estado del Arte

2.1 Introducción

En los próximos subapartados se pretenderá ahondar en la historia de los videojuegos, teniendo en cuenta tanto el aspecto técnico del videojuego como las máquinas que lo han soportado.

Según vayamos avanzando en este estudio se irá mostrando cómo la Inteligencia Artificial ha ido escalando posiciones dentro de las variables a tener en cuenta en la creación de los videojuegos por parte de los productores.

2.2 Los videojuegos

Los videojuegos son juegos electrónicos en donde una o más personas interactúan, por medio de un controlador, con un dispositivo dotado de imágenes de video.

Los primeros videojuegos aparecieron a mediados del siglo pasado. Si se tuviera que precisar, es posible que el primero de todos apareciera en 1947 cuando dos pioneros de las telecomunicaciones, Thomas T. Goldsmith y Estle Ray Mann, decidieron llevar a cabo un experimento con un dispositivo electrónico de simulación. No dejaba de ser una adaptación de un radar (de los usados en los buques armados) con válvulas que se proyectaban sobre una pantalla de rayos catódicos y que era capaz de calcular una curva de lanzamiento de misiles hacia objetivos virtuales. Este experimento se llamó "Lanzamiento de Misiles".

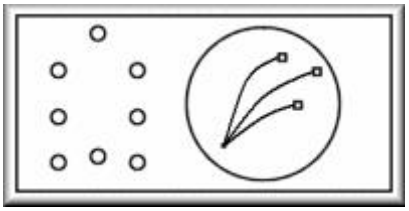

Ilustración 1: Lanzamiento de misiles 1947

Siempre ha habido bastante controversia en torno a si se podía denominar videojuego al “Lanzamiento de misiles” teniendo en cuenta que no contaba con movimiento, ya que dibujaba en la pantalla de rayos catódicos la solución simulada, y que tampoco se podía tratar como juego, puesto que no presentaba una entrada para poder interactuar durante la partida.

Fue el videojuego “OXO” el que despertó esta discusión, ya que fue creado 5 años después por Alexander Sandy Douglas para la universidad de Cambridge, ofreciendo al

usuario poder interactuar durante la partida. Este juego gráfico era una representación del famoso juego "Tres en raya" y la partida se realizaba entre jugador y computadora (que realizaba su movimiento en función del movimiento previo del jugador).

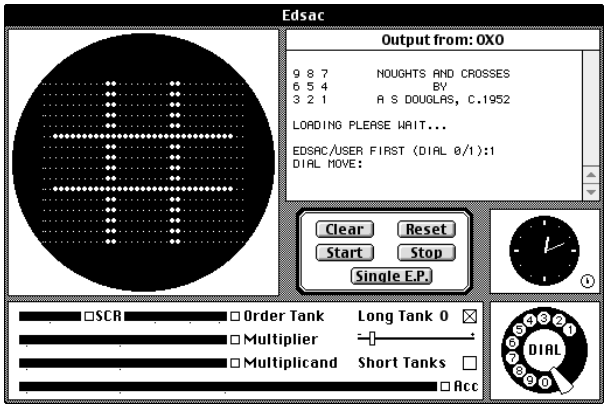

Ilustración 2: OXO en EDSAC

Alexander escribió este videojuego para la computadora EDSAC, el primer calculador electrónico de la historia. Al ser una computadora única creada para la universidad de Cambridge, el juego de OXO para EDSAC no estuvo disponible para el público en general y, aunque estuviese disponible, las dimensiones de la máquina eran tales que sería complejo adecuarlas en una vivienda convencional.

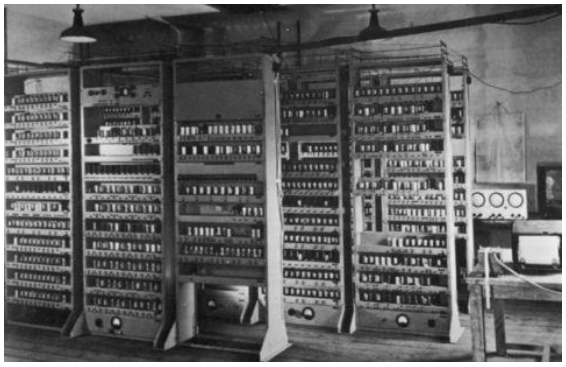

Ilustración 3: EDSAC

En 1957, y con el deseo de entretener a los visitantes del Brookhaven National Laboratory, William Higinbotham creó el juego "Tennis for two" haciendo uso de un osciloscopio conectado a un computador. El objetivo era muy simple: jugar al tenis con dos raquetas (dos líneas móviles) que interceptaban una pelota (punto en oscilación) en la pantalla circular del osciloscopio. Este, presentaba video-animación así como el título de primer videojuego para dos jugadores. El creador, al no patentar este juego, "dejó en bandeja" su explotación comercial a la empresa Atari que, 15 años más tarde, reventó el mercado con su juego "PONG" claramente inspirado en "Tennis for two".

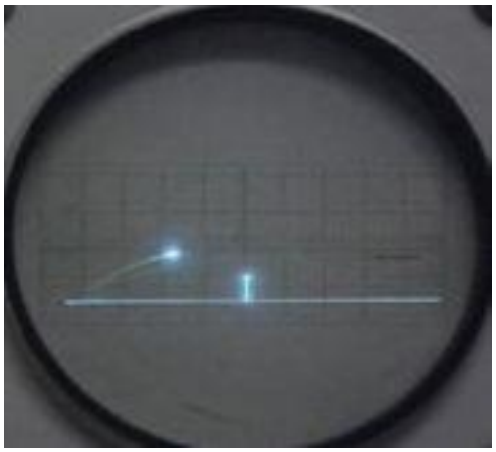

Ilustración 4: Tennis for two

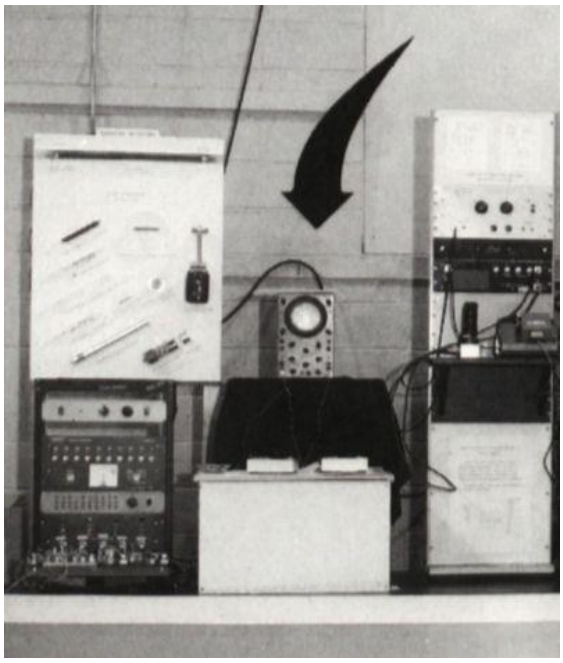

Ilustración 5: Osciloscopio usado para Tennis for two

Más tarde, en 1961, salió a la luz "Spacewar", pudiéndose denominar como el primer videojuego tipo Shooter y el primero en ser trasladado a otras computadoras. El juego fue creado por Stephen Russell y, apoyándose en una máquina PDP-1, permitía la interacción de dos jugadores teniendo que combatir en el espacio exterior con dos naves disparándose entre sí.

Ilustración 6: Spacewar

El éxito de este juego se podría calificar como "local", ya que este tipo de computadoras se podían encontrar únicamente en las universidades pero en la sociedad no era viable ni por espacio ni por el coste asociado.

Ilustración 7: PDP-1

Hasta aquí los videojuegos eran meros experimentos que no salían de los laboratorios de universidades/museos y que el público en general no podía disfrutar. Sin salir de la universidad pero ofreciéndolo a quien quisiera probarlo, y con carácter comercial, en 1971 nació "Galaxy game". Un videojuego creado (basado en Spacewar) por dos estudiantes de Stanford que la instalaron en una pequeña máquina (tipo arcade) y colocaron en una tienda dentro del campus universitario.

La máquina disponía de coin-door (mecanismo para recaudar monedas), de forma que para poder jugar una partida tenías que abonar los 5 centavos que valía. El dinero no parecía ser un impedimento ya que se formaban colas interminables de personas ansiosas por probar este gran invento.

Ilustración 8: Galaxy game

Aunque el "Galaxy game" fuera la primera máquina en explotar económicamente un juego, la primera máquina producida en serie para la comercialización masiva fue la "Computer Space" en 1971. Esta máquina, producida por Nutting Associates, contenía otro variante del mítico juego "Spacewar" y podría denominarse como la primera que salió del ámbito experimental para ofrecerse a la sociedad.

Ilustración 9: Computer Space

Se fabricaron hasta 1000 ejemplares de la "Computer Space" y presentaron algunas versiones para dos jugadores.

Ilustración 10: Máquina Computer Space

Aunque la máquina "Computer Space" fuera la primera en abrirse al público en general, la primera superventas e invasora de hogares fue "PONG", en 1972, de la mano de Atari. Como ya comentamos anteriormente, la falta de patente en el juego "Tennis for two" dejaron en bandeja a la compañía Atari la comercialización y llegada a los hogares de este superclásico.

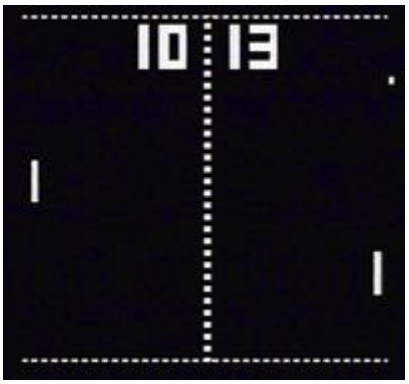

Ilustración 11: PONG

En este punto, la computadora sufrió un cambio drástico (en su volumen sobre todo) de forma que fue más asequible hacerse con videojuegos en los hogares.

Ilustración 12: Atari PONG

A partir de aquí, el público comenzó a considerar a los videojuegos como algo necesario en su vida ociosa y los productores vieron en esto un filón que no ha hecho más que crecer hasta la actualidad mejorando día a día sus apartados gráficos, su inteligencia artificial así como sus guiones.

2.3 La Inteligencia Artificial en los videojuegos

Desde que se crearon las primeras computadoras existió la necesidad de experimentar en busca de la creación de instrucciones que permitieran a la máquina poder decantarse por una u otra opción basándose en el entorno que le rodea.

En cuanto a la Inteligencia Artificial en el mundo de los videojuegos, vamos a encontrarnos con juegos que carecen de inteligencia artificial alguna, véase el Tetris, así como otros que centran gran parte de su potencial en ella, como Los Sims. La gran cantidad de géneros y personajes que hay en el mundo de los videojuegos hace necesario que se considere una interpretación amplia de lo que se puede definir como Inteligencia Artificial.

Además, en algunos casos es posible que lo interesante de un juego es que la Inteligencia Artificial no sea óptima y dependa del personaje en sí, es decir, disponer de personajes con baja Inteligencia Artificial puede hacer del juego una opción atractiva por la versatilidad.

De esta manera, vamos a indagar sobre los tipos de videojuegos, cómo ha ido mejorando la Inteligencia Artificial y cuáles han sido los métodos y técnicas usados para hacer esto posible.

2.3.1 Tipos de videojuegos

Es casi imposible poder clasificar unívocamente a cada videojuego, ya que la mayoría pertenecen a varios tipos, pero una posible clasificación sería la siguiente:

- **Acción:** Los videojuegos de acción tienen un componente básico: eliminar enemigos. De esta manera, podríamos decir que aquellos videojuegos que contengan algún contenido violento pueden ser clasificados como juegos de acción.

Ilustración 13: Jet Set Radio

Dentro de este tipo de clasificación encontraríamos muchas sub-clasificaciones:

  - **Shooter:** El popular género de los "Shooter", ya sean en orientación horizontal o vertical, es aquel en que se controla generalmente una nave espacial (en ocasiones un avión de la 2ª Guerra Mundial, un ser humano volador, un tanque, una escoba mágica con alguien montado, un antivirus informático, una bicicleta, etc.) que dispara proyectiles o energía para destruir naves o criaturas enemigas.

Ilustración 14: Ikaruga

- **FPS:** First Person Shooter, o FPS, es decir, Disparo en Primera Persona, son aquellos juegos donde solamente puede verse el extremo del arma del personaje controlado o sus manos y donde el objetivo es disparar contra los enemigos.

Ilustración 15: Halo 2

- **Beat'em up:** Es la contracción de beat them up (pégales). Son las peleas callejeras en serie, o peleas de un personaje contra varios consecutivos, individual o simultáneamente.

Ilustración 16: Captain Commando

- **Lucha:** Dentro de este subgénero encontraremos la lucha libre, boxeo, combate, lucha en grupo o lucha uno contra uno.

Ilustración 17: Super Street Fighter 2

- **Aventura:** La aventura es un popular género donde el protagonista del juego debe atravesar grandes niveles, luchar contra enemigos y recoger objetos de valor. Normalmente son juegos de larga duración con un argumento extenso y enrevesado.

Ilustración 18: Blade: The edge of darkness

Dentro del género de aventuras encontraremos subgéneros como:

- **Video-aventura:** También se podría incluir como subgénero de acción puesto que aúna lo mejor de la acción con lo mejor de la aventura. Mientras vas moviéndote en una historia extensa, deberás ir sorteando enemigos para llegar al final.

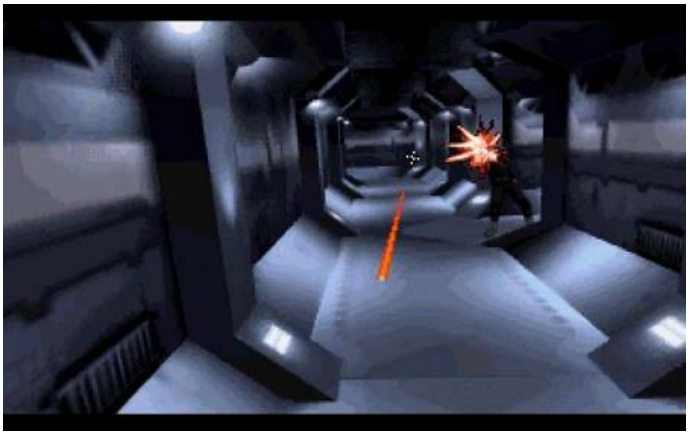

Ilustración 19: Cyberia

- **Aventura Conversacional:** Existen dos tipos dentro de estos juegos pero ambos se caracterizan por mostrar una historia como si de un libro se tratase acompañada de imágenes estáticas y que en determinados momentos dan la opción de intervenir en la historia.

  Por un lado están los solo dan a elegir entre varias opciones fijas que influirán sobre el curso de la aventura, los cuales son

actualmente muy populares en Japón. Se les suele conocer como novelas visuales y son muy limitados en las posibilidades de juego. Un segundo tipo que permite escribir las distintas acciones que se desean realizar durante el juego mediante frases sencillas con verbos en infinitivo. Tuvieron su edad de oro entre finales de los 70 y la década de los 80. Estos en particular fueron los antecesores de las aventuras gráficas.

Ilustración 20: El quijote

- **Aventura gráfica:** Al igual que las aventuras conversacionales, la historia cumple un importante papel en el juego, donde además tendremos un repertorio de acciones muy similar al de las aventuras conversacionales clásicas, pero con la diferencia de poder interactuar con el escenario y los personajes de una forma más gráfica que en las antiguas aventuras conversacionales.

  Generalmente el sistema se basa en combinar acciones, objetos y elementos del escenario, ya sea por ejemplo seleccionando la opción "abrir" y señalando la puerta en el dibujo que aparece en la pantalla para lograr abrirla.

Ilustración 21: Monkey Island 2

- **Simulación:** Los simuladores imitan, emulan o reproducen con desigual exactitud el mundo real en cualquiera de sus facetas: deportes, vuelo, guerra, carreras automovilísticas, mascotas, el cuerpo humano, etc. Dentro de este género encontramos:
    - **God Simulator:** Simulación o representación de un mundo, región, ciudad u otro lugar donde el jugador adopta el papel de un dios o entidad con poderes divinos que controla, ayuda, castiga o encamina a los habitantes de ese mundo o interfiere en su vida cotidiana. El papel del "dios" o jugador es invisible a los ojos de las personas o criaturas del juego, salvo contadas excepciones.

Ilustración 22: Black & White

    - **Mascotas Virtuales:** El objetivo de estos juegos es criar y cuidar a un animal o una criatura, desde que nace, teniendo que

preocuparse por alimentarle, curarle, mimarle, asegurarse de que hace ejercicio y de que duerme lo suficiente, y prácticamente de llevar a cabo todas las tareas propias de hacer de "madre" de la susodicha criatura.

Ilustración 23: Viva Piñata

- **Simulador social:** Los videojuegos de simulación de vida (también conocidos como videojuegos de vida artificial) son un subgénero de los videojuegos de simulación en los que el jugador vive o controla una o más formas de vida artificial. Un videojuego de simulación de vida puede girar en torno a individuos y relaciones, o puede ser una simulación de un ecosistema.

Ilustración 24: Los sims

- **RPG:** El acrónimo RPG significa Rol Playing Game, es decir, juego de Rol. El jugador adopta el papel de un personaje que puede avanzar,

evolucionar y cambiar en un mundo con enemigos, tiendas, objetos para vender o usar y ataques especiales.

Normalmente en el género de los RPG los personajes controlados por el jugador pueden acumular puntos de experiencia derrotando enemigos y subir de nivel cuando dichos puntos alcanzan una cantidad determinada (por ejemplo 100). En los comercios o tiendas ficticios de los juegos pueden adquirirse, vender o intercambiar toda clase de objetos, como pociones para recuperar la salud, comida, artilugios que efectúan un hechizo, aprender golpes o ataques nuevos, medicinas para sanar los envenenamientos, ropa que aumenta la resistencia, etc.

Se subdivide en:

- **Action RPG:** Son juegos de Rol donde la acción y los ataques en tiempo real cobran una mayor relevancia sobre el resto de elementos (como la interacción con otros personajes o la exploración). Durante los combates, el jugador debe reaccionar rápidamente y dar muestras de unos reflejos veloces para vencer, en lugar de la clásica meditación y atributos de los personajes, reinantes en los RPG clásicos.

Ilustración 25: Fable

- **Strategic RPG:** Se caracteriza por ser juegos de rol táctico en donde se juega y ataca por turnos.

Ilustración 26: Shinning Force

- **MMORPG:** Acrónimo de Multiplayer Massive Online Rol Playing Game (Juego de Rol Multijugador Masivo Online). Un género relativamente moderno, ya que es necesaria una velocidad mínima de conexión a Internet.

Ilustración 27: WOW

- **Estrategia:** Son aquellos juegos o entretenimientos en los que, el factor de la inteligencia, habilidades técnicas y planificación y despliegue, pueden hacer predominar o impulsar al jugador hacia la victoria del juego. Los jugadores pueden representar el papel de un empresario, un jefe de estado, un general, o cualquier otro personaje, en los que tendrán que desarrollar una serie de estrategias, gestionando los recursos de los que se dispone, para ganar una batalla, conseguir dinero o puntos, determinada posición, etc., y así conseguir el objetivo final.

Ilustración 28: Age of Empires 3

2.3.2 Evolución de la Inteligencia Artificial en los videojuegos

Aunque ya hemos hablado sobre los primeros videojuegos en apartados anteriores, estos no disponían de inteligencia artificial propia ya que, o bien, eran de 2 jugadores o cualquier acción de un jugador ficticio era programada sin opción de poder tomar una de entre varias posibilidades.

El uso de Inteligencia Artificial aplicada en juegos empezó a dar sus frutos a finales de los 50 gracias a juegos computacionales como las damas o el ajedrez en donde la computadora iba aprendiendo a medida que jugaba y adquiría la forma de atacar o defender para ganar la partida.

Teniendo en cuenta que los primeros videojuegos eran muy básicos (entre otras cosas por la limitación en hardware y recursos), estos contaban con una Inteligencia Artificial poco compleja, basada en algoritmos, y aplicados para resolver situaciones sencillas.

Con la llegada de las primeras videoconsolas, que no dejaban de ser otra cosa sino computadoras menos potentes exclusivas para jugar, los videojuegos fueron incorporando mejoras (pocas) en la Inteligencia Artificial aunque se centraban más en otros aspectos distintos a esta materia.

"Pong" sería un ejemplo de videojuego con limitada Inteligencia Artificial. Esta inteligencia residiría en la paleta del oponente, y no tanto en buscar un modo de

aprendizaje, ya que dependiendo de la dificultad seleccionada, esta paleta golpearía más o menos fuerte o incluso tendría una probabilidad mayor o menor a la hora de errar el golpeo.

La introducción de consolas como NES o SEGA GENESIS con juegos del tipo "Kung Foo" o "Mortal Kombat" respectivamente no añadió mejoras significativas en el aspecto inteligente. Los movimientos de los oponentes estaban condicionados a dónde estaba localizado el jugador o qué estaba haciendo en ese momento.

Ilustración 29: Kung Fuu

Esto fue así hasta que llegó el boom de los videojuegos allá por finales de la década de los 90 en donde los ordenadores personales (con memorias y procesadores ampliados) comenzaron a estar en casi todos los hogares junto con videoconsolas más potentes obligando a los productores conseguir no solo una mejora en la calidad gráfica o en los guiones, sino en ofrecer al consumidor productos con una Inteligencia Artificial que le dieran ese "plus" frente a sus competidores.

La Inteligencia Artificial pasó de dotar de cierta habilidad a un personaje en concreto del videojuego a intentar simular el comportamiento habitual de cada uno de los individuos presentes en el escenario y decidir qué acción tomar en base a cómo estén sucediendo los acontecimientos.

Un claro ejemplo de ello es el juego de "Los Sims", un videojuego de simulación social y estrategia publicado en el año 2000, en donde cada sim tiene personalidad propia y se controla de forma individual; tanto es así que incluso dependiendo del personaje que actué, a la hora de ir al retrete subirá o no la tapa.

Ilustración 30: Los Sims

A día de hoy, viendo el potencial que disponemos en cuanto a hardware se refiere, no observamos grandes mejoras de una generación de procesadores o videoconsolas desde el punto de vista gráfico pudiendo decir que se está llegando a la cima en ese apartado. Por esta razón, la industria del videojuego se está centrando tanto en mejorar la Inteligencia Artificial y es, en estos momentos, uno de los pilares principales a la hora de encumbrar o desechar un videojuego.

2.3.3 Métodos y técnicas usadas para simular inteligencia

Para poder simular inteligencia es imprescindible poder dotar al sistema de determinados algoritmos que ayuden a abstraer los problemas a resolver de forma que pueda obtener la solución precisa y, siempre que se pueda, hacerlo de la mejor manera posible encontrando la mejor solución o desplegando todas las posibles soluciones.

A continuación indicaremos las distintas técnicas usadas para conseguir simular inteligencia.

2.3.3.1 Sistemas de Producción

En general, podemos afirmar que un problema consiste en una situación inicial desde la que se parte, una meta a la que queremos llegar y el conjunto de los medios que tenemos para poder llegar con éxito al objetivo.

Si intentamos orientar la definición anterior al campo de la informática, lo que se pretende a partir de un problema es construir un sistema que lo resuelva. De esta manera, si decimos que un problema consta de situación inicial de la que se parte nos estaremos refiriendo a los datos de entrada, la meta a la que queremos llegar corresponderá a la

salida computacional y el conjunto de medios serán las acciones que irán transformando el estado inicial al estado final.

Un Sistema de Producción intenta generalizar la anterior aproximación constando de:

- **Base de Hechos:** Son los predicados que describen el problema concreto. Se podría definir como "todo aquello que en esos momentos es verdadero".
- **Base de Reglas:** Son reglas que describen los mecanismos de razonamiento permitiendo resolver los problemas.
- **Motor de Inferencia:** Es el encargado de determinar el método de razonamiento utilizando estrategias de búsqueda y resolviendo los conflictos.

El funcionamiento del Sistema de Producción consiste en realizar ciclos buscando reglas que puedan ser cumplidas a partir de los hechos que existen en ese ciclo. En cada ciclo, el motor determina aquel subconjunto de reglas cuyas precondiciones son satisfechas con los contenidos actuales de la base de hechos. Una vez el sistema tiene el subconjunto de reglas, el motor de inferencia se encarga de seleccionar aquella regla a ejecutar teniendo en cuenta los criterios de selección en caso de conflicto (prioridad más alta, aleatoriedad, primera regla en cumplir el "matching"…) y una vez seleccionada la regla se lleva a cabo la postcondición de la misma ejecutando las acciones pertinentes y actualizando la memoria del trabajo o Base de Hechos, si procede, con nuevos hechos o eliminando aquellos que se determinen en la regla.

2.3.3.2 Algoritmos de búsqueda

En el apartado anterior, indicábamos que un problema constaba de un estado inicial, un estado final y las acciones disponibles que disponíamos para llegar de un estado a otro. No hacíamos mención a la posibilidad de que para ir pasando de un estado a otro (con estados intermedios) podíamos encontrarnos con un coste asociado. Si introdujéramos esta nueva variable en la ecuación, quizás ya no sería interesante que la elección de una regla a ejecutar estuviera basada en la aleatoriedad o prioridad por pesaje sino aquella que nos costase menos de cara a llegar a un camino óptimo a la solución mediante una búsqueda previa.

Los algoritmos de búsqueda son una de las técnicas más conocidas en Inteligencia Artificial. Un algoritmo de búsqueda se puede representar como el recorrido por un árbol

(o grafo) en el que cada nodo representa un estado y cada rama las relaciones entre los estados de los nodos conectados. Una solución al problema será la ruta desde el punto inicial al estado final y la calidad de esa solución estará marcada por el coste empleado en la ruta.

Dependiendo de sus características, encontramos tres tipos de búsquedas, la búsqueda no informada, la búsqueda informada (o heurística) y la búsqueda de varios agentes.

2.3.3.2.1 Búsqueda No Informada

Como su propio nombre indica, la Búsqueda No Informada es aquella que no dispone de información de por dónde se debe realizar la búsqueda. Por esto, existen algunos métodos de la teoría de grafos que nos permitirán hacer uso de esta búsqueda. A continuación indicaremos algunos de ellos.

- Búsqueda en amplitud: Este algoritmo va recorriendo el grafo desde el nodo inicial explorando todos los vecinos de ese nodo. Una vez realizado esto, procede a explorar para cada uno de los vecinos sus respectivos vecinos adyacentes hasta que recorre todo el árbol. De esta manera se podría decir que va recorriendo el árbol por niveles funcionando como el concepto informático de cola en donde los primeros nodos generados son los primeros en salir o "First-in, First-out" (FIFO).

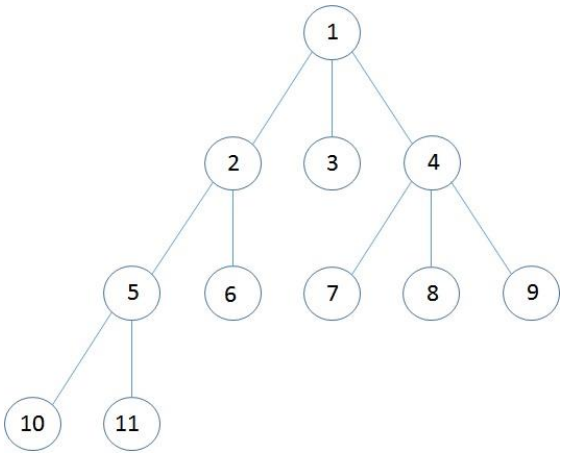

Ilustración 31: Búsqueda en Amplitud

- Búsqueda en profundidad: Este algoritmo va recorriendo el grafo desde el nodo inicial eligiendo el primer camino siempre de manera que va profundizando por una rama. Una vez llegado al final de la rama, o si ha indicado una profundidad máxima, regresa ("back-tracking") al nodo padre e intenta la búsqueda por otro

camino. Esta estrategia correspondería en el mundo computacional al modo en que funcionan las pilas, "Last-in, First-out" (LIFO).

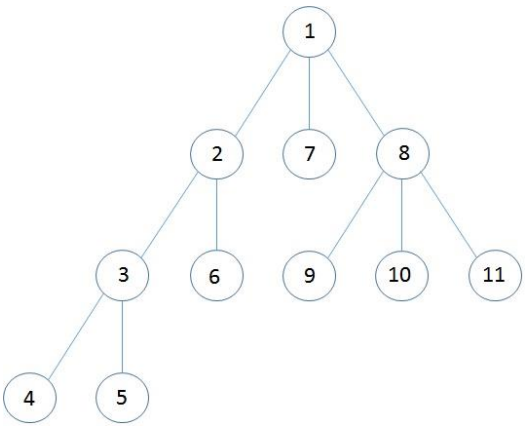

Ilustración 32: Búsqueda en Profundidad

2.3.3.2.2 Búsqueda Heurística

La Búsqueda Heurística o Búsqueda Informada se basa en información detallada que nos permite elegir aquellos caminos más prometedores en pos de llegar a la meta deseada. Newell, Shaw y Simon en 1963 dieron la siguiente definición: "Un proceso que puede resolver un problema dado, pero que no ofrece ninguna garantía de que lo hará, se llama una heurística para ese problema".

Dentro de este tipo de búsquedas, podemos encontrar diversos métodos de los cuales citaremos algunos.

- Técnica de Escalada: Es una evolución de la técnica de búsqueda por profundidad en la que, a diferencia de la de profundidad, por cada nodo dispondremos de una forma de evaluar cómo está de lejos o cerca la solución. La forma más común en estos casos es la de función de evaluación.

  Un ejemplo claro de esta técnica se encuentra en el juego 8-Puzzle, cuya función de evaluación se podría definir como:

  $$f(nodo) = nº - casillas - bien - colocadas(nodo)$$

  De esta manera, la decisión de tomar una elección entre varios estados estará basada en el valor máximo del resultado de esa función.

  Esta técnica no asegura que se alcance la solución óptima y puede arrojar problemas cuando la función de evaluación seleccionada es maximizante y la evaluación de los sucesores de un nodo es menor o igual que la del nodo. Para paliar estos problemas, se podría retroceder o, en lugar de hacer "back-tracking", dar un paso más para buscar los sucesores de los sucesores del nodo a evaluar.

- Técnicas de mejor-primero: Esta técnica selecciona un nodo para la expansión teniendo en cuenta una función de evaluación. Usa una función $f(n)$ para cada nodo, que aún no haya sido expandido, con la intención de localizar aquel que sea más prometedor y expandirlo.
- A^*: El algoritmo A^* es el procedimiento más conocido dentro de las técnicas de mejor-primero. Sigue empleando una función heurística con la idea de no expandir aquellos trayectos cuyos costes ya se saben que son caros. La función que se intenta minimizar sería:

  $$f(n) = g(n) + h(n)$$

  donde nos encontramos una nueva función, $g(n)$, resultando ser el coste estimado de ir desde la raíz al nodo n. De esta manera, conforme se van calculando otros nodos, se podrán localizar caminos menos costosos para llegar de la raíz al nodo n por lo que este coste se puede tomar como estimado. Denominaremos $g^*(n)$ al coste real del camino menos costoso que habría del nodo raíz a n pero sólo se sabrá una vez hayamos encontrado todos los nodos del camino solución.

  Si queremos obtener la estimación del coste que tendrá desde el nodo n hasta el final estaremos hablamos de $h(n)$ y $h^*(n)$ será el coste mínimo real.

  Lo que hace tan conocido al algoritmo A^* es que encontrará (siempre que exista) la solución al problema y, además, de haber una solución óptima también dará con ella.

2.3.3.2.3 Búsqueda con varios agentes

Hay ciertos tipos de situaciones en las que es preciso hacer uso de métodos de búsqueda distintos a los que hemos comentado hasta ahora. Este tipo de problemas son aquellos en los que hay varios agentes, o jugadores, en donde se van realizando acciones por turnos.

Al realizar una acción por turno, el contrincante dispondrá en todo momento de toda la información necesaria del contrario para llevar a cabo su acción.

Ejemplos claros de este tipo de búsquedas serán los algoritmos MiniMax y Poda Alfa-Beta.

- MiniMax: Lo que se busca con este método es intentar maximizar la evaluación de las jugadas del jugador teniendo en cuenta que el contrario buscará minimizarlas todas.

Para ello, se hace uso de una función ("función estática") que analiza todas las situaciones dando un valor.

La forma de usar la función será, primero, desarrollando una búsqueda por niveles hasta que se generen los nodos de un cierto nivel y, segundo, aplicando dicha función para llegar a tres posibles situaciones:

- La jugada sale ganadora para el jugador maximizante (resultado positivo).
- La jugada sale perdedora para el jugador minimizante (resultado negativo).
- La jugada no es ganadora para ningún jugador.

- Poda Alfa-Beta: Algoritmo con los mismos valores producidos que el MiniMax pero con la mejora de que ignora aquellas acciones que son incapaces de mejorar otras ya conocidas.

  Se basa en la idea de disponer dos umbrales, alfa y beta, que representen el valor de cota inferior y superior respectivamente dentro de una ventana a la cual deben pertenecer los valores de la función de evaluación para que sean considerados. En los nodos MAX se debe maximizar el valor de los nodos sucesores haciendo uso del parámetro α y en los nodos MIN se minimizará el valor con el parámetro β.

  La poda en los nodos MAX se lleva a cabo si el α de un nodo MAX en una profundidad p, es mayor o igual que la β del nodo padre.

  La poda en los nodos MIN se lleva a cabo si el β de un nodo MIN en una profundidad p, es menor o igual que el α del nodo padre.

2.3.3.3 Redes neuronales

Las redes neuronales buscan la solución de problemas complejos, no mediante una secuencia de pasos, sino inspirándose en el cerebro humano haciendo uso de la combinación simple de procesos (neuronas) interconectados para conseguir resolver problemas relacionados con el reconocimiento de formas, predicción, codificación, control y optimización entre otras aplicaciones.

Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida. Esta salida viene dada por tres funciones:

- Función de propagación o excitación: Consiste en el sumatorio de cada entrada multiplicada por el peso de su interconexión (valor neto). Si el peso es positivo, la conexión se denomina excitatoria; si es negativo, se denomina inhibitoria.

- Función de activación: Actúa modificando a la anterior. Puede no existir, siendo en este caso la salida la misma función de propagación.
- Función de transferencia: Aplica al valor devuelto por la función de activación. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que queramos darle a dichas salidas. Algunas de las más utilizadas son la función sigmoidea (para obtener valores en el intervalo [0,1]) y la tangente hiperbólica (para obtener valores en el intervalo [-1,1]).

Las redes neuronales artificiales tienen muchas ventajas debido a que están basadas en la estructura del sistema nervioso. Algunas son:

- Aprendizaje: Tienen la habilidad de aprender mediante una etapa que se llama etapa de aprendizaje. Esta consiste en proporcionar a la red neuronal datos como entrada a su vez que se le indica cuál es la salida (respuesta) esperada.
- Auto organización: Una red neuronal crea su propia representación de la información en su interior, quitándole esta tarea al usuario.
- Tolerancia a fallos: Al almacenar la información de forma redundante, la red neuronal puede seguir respondiendo de manera aceptable aun si se daña parcialmente.
- Flexibilidad: Puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada.
- Tiempo real: La estructura de una red neuronal es paralela, por lo cual si esto es implementado con computadoras o en dispositivos electrónicos especiales, se pueden obtener respuestas en tiempo real.

2.3.3.4 Aprendizaje automático

El Aprendizaje Automático es una rama de la Inteligencia Artificial que abarca diferentes técnicas que permiten dotar a los computadores de la capacidad de "aprender" modelos computacionales tales que, de forma automática, les permitan resolver nuevos problemas o mejorar su comportamiento en problemas ya vistos.

Existen distintas formas de clasificar todas las técnicas pertenecientes al ámbito del aprendizaje automático. Así, atendiendo a su naturaleza inferencial, se puede hablar de técnicas de aprendizaje inductivas, deductivas, abductivas y por analogía. Atendiendo al tipo de modelo aprendido, se habla de técnicas simbólicas (modelos que manejan sólo conocimiento expresado en forma simbólica), conexionistas (si el conocimiento es sólo

de tipo numérico) y mixtas (modelos que participan de los dos tipos de conocimiento anteriores). Finalmente, dependiendo de si en el conjunto de datos de entrenamiento existe o no información de la clase o concepto al que pertenece cada ejemplo, se habla, respectivamente, de técnicas de aprendizaje supervisado y no-supervisado.

3. Objetivos del Proyecto Final de Carrera

El objetivo principal del Proyecto Final de Carrera es dotar a cada sim de una historia personal que sea lo más variada posible y acorde a una serie de factores propios del individuo. *"La vida es simplemente un mal cuarto de hora formado por momentos exquisitos"* (Oscar Wilde). Esos momentos exquisitos serán eventos que a cada persona le van ocurriendo a lo largo de su vida y tendrán lógica al final de la misma. El otorgar una historia pasada a cada personaje va a dar un mayor realismo al juego asemejándolo a la vida real. Todos los actores tienen una serie de factores propios que los distinguen del resto tales como su personalidad, sus habilidades, sus gustos o incluso su azar por lo que los eventos que les pueden ocurrir a cada uno de ellos tienen una relación directa a cómo es la persona.

De esta manera, siendo el objetivo principal la generación y posterior modificación de historias de personajes de videojuegos podemos indicar como objetivos secundarios los que se enumeran a continuación:

- Aumento y modificación de la ontología.
- Creación de nuevos personajes con características dispares entre sí.
- Integración con AI-LIVE haciendo uso de habilidades, personalidad y características de personajes.
- Interacción entre personajes gracias a las historias creadas.
- Traspaso de conocimiento entre personajes mediante eventos pasados de historia de tipo habilidad.
- Generación de narrativa de la historia de personajes.
- Almacenaje y recuperación de historias de personajes.

Así pues, vamos a dotar al juego AI-LIVE de un bien necesario en todas las sociedades: LA HISTORIA PASADA DE UNA PERSONA.

4. Memoria del trabajo realizado

4.1 Introducción

En este punto de la memoria se va a describir todo el trabajo que se ha añadido al juego AI-LIVE. El proyecto aporta una nueva funcionalidad al mismo, la generación automática de historias para cada actor.

Hasta la fecha anterior a este proyecto, el juego AI-LIVE permitía a un actor poder realizar acciones (unas consumían más tiempo que otras) basándose en secuencias lógicas con los gustos o estados de ánimos como influencia. Estas acciones podían corresponder a necesidades básicas a cubrir (comer, descansar, ducharse, etc.), acciones intermedias para poder llegar a realizar (tras varios turnos), alguna acción final o acciones que supongan un desarrollo personal para el propio actor como el aprendizaje de una habilidad.

En este proyecto se ha decidido abrir una nueva vía hacia la búsqueda de la semejanza del comportamiento humano en AI-LIVE, dotar a todo actor de una historia propia y personal que tenga lógica con sus gustos, habilidades o personalidad.

Para ello, y al tratarse de algo nuevo en AI-LIVE, decidí crear un módulo aparte que me permitiera gestionar de manera más coherente y sencilla esta nueva funcionalidad. Se han tenido que modificar clases ya existentes así como crear nuevas clases, funciones, métodos y reglas que permitieran desarrollar el proyecto de manera correcta.

Adicionalmente, se ha creado la acción de poder enseñar a otro actor una habilidad a partir de un evento pasado de historia abriendo nuevas opciones a la mejora de habilidades creada ya anteriormente [Blanco, 2013].

4.2 Arquitectura de la aplicación

AI-LIVE está basado en una arquitectura de tipo cliente-servidor en donde los distintos clientes, ya sean manuales o de Inteligencia Artificial, realizan acciones y se conectan al servidor, que es el que gestiona las acciones que cada cliente ha realizado y actualiza los estados y el escenario sobre el que transcurre el juego.

La comunicación entre estos se hace gracias a sockets sobre el protocolo TCP/IP ya que facilita la división de procesos entre los clientes y la gestión de toda la información está en un único servidor. El hecho de que toda la información esté en el servidor permite que distintos clientes se puedan conectar y obtener el estado actual del escenario y lo que

se permite hacer en el mismo tras los distintos turnos de clientes.

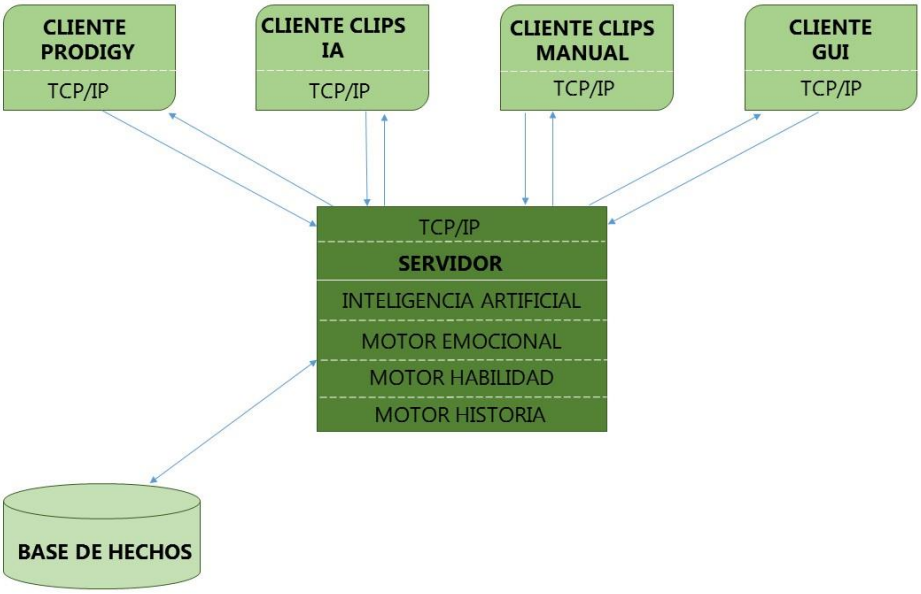

Ilustración 33: Arquitectura AI-LIVE

Tal y como se muestra en la ilustración 33, los clientes realizarán las peticiones de las acciones a llevar a cabo al servidor, este último validará si esa acción es viable y, en caso de serlo, actualizará el estado.

4.3 Módulos de la aplicación

4.3.1 Servidor

El servidor se podría considerar como el núcleo de AI-LIVE ya que es el punto central y principal para todos los clientes que se vayan conectando. Es el encargado de gestionar los turnos en los clientes de forma que todos puedan realizar su acción en su debido momento y actualiza el estado del cliente y del escenario para el resto de clientes. El servidor está compuesto por dos módulos:

- Principal: Establece las conexiones de clientes y asigna los turnos de los mismos. En caso de que llegue un nuevo cliente, éste se encarga de conectarle y asignarle el turno que le corresponda. Además, realizará las llamadas al módulo de Inteligencia Artificial del servidor.
- Inteligencia Artificial: Implementado en CLIPS, sobre él recae la responsabilidad de ejecutar (siempre y cuando sea viable) todas las acciones solicitadas por el cliente y mantener el estado de AI-LIVE actualizado. En este módulo se encuentra

el estado del escenario del juego que será modificado tras cada turno de cliente. Debido a las sucesivas funcionalidades que se han ido añadiendo en este módulo, se han ido creando nuevos ficheros o motores que indicaremos a continuación:

- Emocional: Se encuentran las acciones relativas a las emociones (hablar, cotillear…) que solicita realizar el cliente al servidor.
- Habilidad: Contiene las acciones relativas al aprendizaje que son solicitadas desde el cliente.
- Historia: Último motor en ser añadido al proyecto AI-LIVE. Es el encargado de crear la historia del cliente que se acaba de conectar al servidor y actualizará las historias de aquellos clientes ya creados que compartan algún evento nuevo con el cliente recién incorporado al juego.

4.3.2 Clientes

Dentro del juego AI-LIVE siempre vamos a encontrar un único servidor, pero en el caso del cliente, habrá tantos como se conecten. El cliente se conectará, siendo asignado por el servidor, a un socket para que pueda realizar las peticiones de acciones al núcleo o servidor.

El cliente tendrá dos módulos para gestionar los 4 posibles clientes que hay en AI-LIVE. Estos módulos serán:

- Principal: Es el responsable de que exista la conexión desde el cliente con el servidor y de que la comunicación entre ambos extremos sea la correcta y llegue íntegra.
- Ejecución: Se encarga de realizar la funcionalidad específica de cada cliente. Actualmente hay 4 tipos de clientes:
  - Prodigy: Este tipo de cliente se basa en Planificación Automática y, haciendo uso de un planificador, es capaz de seleccionar aquella tarea que mejor convenga al actor controlado.
  - Clips: Cliente que se apoya en un sistema de producción basados en reglas que elegirá aquella tarea que se ajuste a las reglas y prioridades del actor.
  - Clips Manual: Al igual que el cliente anterior, se apoya en un sistema de producción basado en reglas pero, en este caso, permitirá al usuario elegir la acción a ejecutar según el estado del actor y el escenario.
  - GUI: Cliente de interfaz gráfica de usuario que representa en 3D el escenario del juego AI-LIVE.

4.4 Comunicación Cliente-Servidor

Con el fin de seguir explicando cómo funciona AI-LIVE, este apartado se centrará en la forma en la que se comunican todos los clientes con el servidor.

4.4.1 Etiquetas

Para que el servidor intercambie mensajes con un cliente cualquiera, y viceversa, es necesario que haya un protocolo de comunicación basado en etiquetas que ambas partes sean capaces de interpretar. Las etiquetas usadas para la comunicación serán:

- HOLA: Se trata de la etiqueta que inicia la conexión entre los módulos servidor y cliente de la aplicación. Junto a la etiqueta se envía la información relativa a versión y opciones soportadas en la comunicación.
- STAG: Etiqueta que usan los clientes para informar al servidor del escenario desde el que comenzarán en el juego.
- ACTR: Etiqueta que es usada por los clientes para controlar el id de cada actor que se encuentra en el juego.
- ACTN: Etiqueta que es usada por los clientes para identificar la acción que va a ser realizada, por actor, en el servidor.
- STAT: Esta etiqueta es empleada por el servidor para enviar el estado actual del juego al cliente destino.
- GO: Etiqueta exclusiva para comunicación entre servidor y cliente GUI para indicar un módulo al otro que ha finalizado la actualización de información y que el otro módulo puede continuar con su ejecución.
- EX: Etiqueta usada por cliente GUI que indica al servidor que ha finalizado su ejecución.

4.4.2 Pasos a seguir para el protocolo de comunicación

Teniendo claro cuáles son las etiquetas usadas entre los módulos, vamos a explicar el orden que sigue cada módulo para comenzar una conexión, y mantenerla, entre el servidor y los clientes mediante el protocolo de comunicación.

4.4.2.1 Pasos a seguir por el servidor

1. Arranca motor Inteligencia Artificial del servidor.
2. Arranca el modo escucha de red para esperar a clientes que se conecten.
3. Una vez recibe el mensaje de conexión entrante por parte de cliente, establece la conexión.

4. Loop en ejecución:
    a. Selección de cliente en turno asignado.
    b. Envío estado actual al cliente y al GUI (en caso de que esté seleccionado).
    c. Recepción acción a realizar por parte de cliente.
    d. Envío de acción a realizar por parte de cliente al servidor de Inteligencia Artificial.
    e. Recepción del estado que devuelve el servidor de Inteligencia Artificial.

4.4.2.2 Pasos a seguir por el cliente CLIPS o CLIPS Manual

1. Envío de mensaje de conexión al servidor.
2. Conexión establecida con servidor.
3. Loop en ejecución:
    a. Recepción del estado actual que nos envía el servidor.
    b. Interpretación del estado recibido.
    c. Elección de la acción a realizar según el cliente. Si es CLIPS la acción será la que esté más priorizada o cumpla con las condiciones por delante del resto y si es CLIPS Manual la acción la seleccionará el usuario.
    d. Envío de acción a realizar al servidor para que la verifique y, en caso de ser posible, la ejecute.

4.4.2.3 Pasos a seguir por el cliente GUI

1. Envío de mensaje de conexión al servidor.
2. Conexión establecida con servidor.
3. Loop en ejecución:
    a. Recepción del estado de los objetos gráficos.
    b. Representación gráfica del estado recibido.

4.5 Modelo de conocimiento

El juego AI-Live es el resultado de todos los aportes anteriores de diferentes proyectos de fin de carrera que han ido modificando y añadiendo modelo de conocimiento. Todas las clases están recogidas en una ontología del sistema para poder desarrollar las nuevas funcionalidades.

La mayoría de la ontología está albergada en el archivo "ontology.clp". Sin embargo, la parte específica relacionada con las habilidades y capacidades de los personajes, se encuentra en el archivo "hability_ontology.clp" y la parte relativa a la historia del sim, que sería el grueso de mi proyecto, está localizada en "story_ontology.clp".

A continuación se mostrará de manera gráfica el modelo de conocimiento con la posterior explicación en los aportes y cambios realizados.

4.5.1 Aportes al modelo de conocimiento en "ontology.clp"

Para poder abarcar el proyecto de Generación Automática de Historias ha sido necesario crear dos clases en el fichero principal de "ontology.clp". Estas nuevas clases corresponden a acciones que van a permitir hacer uso de las historias creadas en un actor.

Lo que se muestra a continuación son las clases que representan acciones, las que están en color negro son las que tenían en un origen el fichero, las que están en verde son las anteriores al proyecto relativas a las habilidades y las rojas son las propias del proyecto de Generación Automática de Historias.

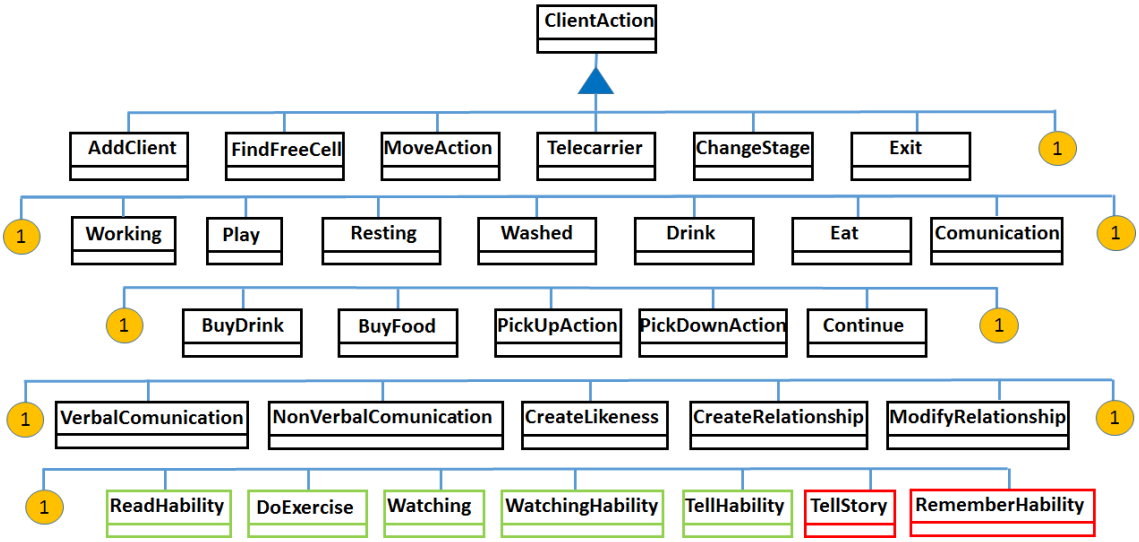

Ilustración 34: Acciones Disponibles en AI-LIVE

- TellStory: Clase de tipo VerbalCommunication creada para poder realizar la acción de contar la historia de un sim a otro.
- RememberHability: Clase de tipo VerbalCommunication creada para poder realizar la acción de transmitir conocimiento entre dos actores a través de algún evento de habilidad presente en la historia de uno de los dos sims. Esta acción está

vinculada al módulo de aprendizaje de habilidades y al de historia. El único atributo propio que compone esta clase es:

- Subject: Instancia que hace referencia al evento de habilidad presente en la historia.

4.5.2 Cambios en el modelo del conocimiento

Para poder hacer práctica la nueva funcionalidad ha sido necesario añadir a clases ya existentes en "ontology.clp" nuevos atributos que pudieran recoger aquellas partes de la Generación Automática de Historias.

A continuación indicaremos las clases afectadas y los nuevos atributos añadidos:

- AddClient: Clase de tipo ClientAction necesaria para la creación de nuevos actores al universo AI-LIVE. Ha sido añadido un nuevo campo:
  - GenerLike: Multicampo de tipo SYMBOL que hace referencia a los gustos del actor en general y no a los objetos que hay en el universo AI-LIVE.

- Actor: Clase de tipo Entity que identifica los valores y atributos de cada sim. Se nutre, en parte, de la clase AddClient. Los campos añadidos son:
  - Age: Campo de tipo NUMBER que hace referencia a la edad que tiene el sim a la hora de ser añadido al juego. Se nutre del campo con mismo nombre de la clase AddClient.
  - SentimentalSituation: Campo de tipo SYMBOL que nos indica la situación emocional que tiene el sim en cada momento. Los posibles valores válidos serían:
    - Single: Se toma este valor como defecto al principio del juego, comenzando el personaje siendo soltero, durante la generación de la historia puede cambiar.
    - Married: Si, en su generación de la historia o de otro sim, hay un evento de este sim y otro que indica que se han casado tomará este valor y no permitirá que se vuelva a casar a menos que se divorcie antes.
    - Divorced: Puede ser que, tras haberse casado un sim con otro, haya eventos de discusiones entre ellos que haga terminar su relación en divorcio.

- Story: Multicampo de INSTANCIAS de clase tipo Event usadas para que cada actor tenga su historia con todos los eventos ocurridos en su vida. Está en orden cronológico y puede verse aumentado con la creación de nuevos actores si tienen eventos en común.
- Storied: Campo tipo SYMBOL que hará de control para diferenciar aquel actor que tenga su historia inicial creada del que no. Solo admite dos valores:
  - YES: El actor ya tiene su historia inicial creada.
  - NO: El actor acaba de entrar en el juego y tiene pendiente que su historia sea creada. Valor por defecto.
- ActorType: Campo tipo SYMBOL que diferencia aquellos actores que están en el universo de AI-LIVE de aquellos que son usados para poder tener eventos con actores reales en las historias. Esto es necesario ya que el primer actor que entra en AI-LIVE no podría tener historias entrelazadas con otros actores al no haber ninguno. Dos posibles valores:
  - Playable: Actor real en el universo AI-LIVE. Valor por defecto para actores reales.
  - Fictional: Actor no real usado para relacionarse con otros actores reales de manera que puedan tener una historia más rica.
- GenLike: Multicampo tipo SYMBOL que hace referencia al campo GenerLike de la clase AddClient que recoge los gustos en general del actor.

4.5.3 Aportes al modelo del conocimiento en "story_ontology.clp"

Debido a que este proyecto es una funcionalidad que no comparte ninguna afinidad con el resto de funcionalidades ni con la lógica inicial se ha creído conveniente crear un fichero independiente para abarcar las clases propias a la funcionalidad de la generación de historias. De igual manera, se muestra la clase, en rojo, que se ha añadido para poder satisfacer la funcionalidad de la historia en la entidad Actor.

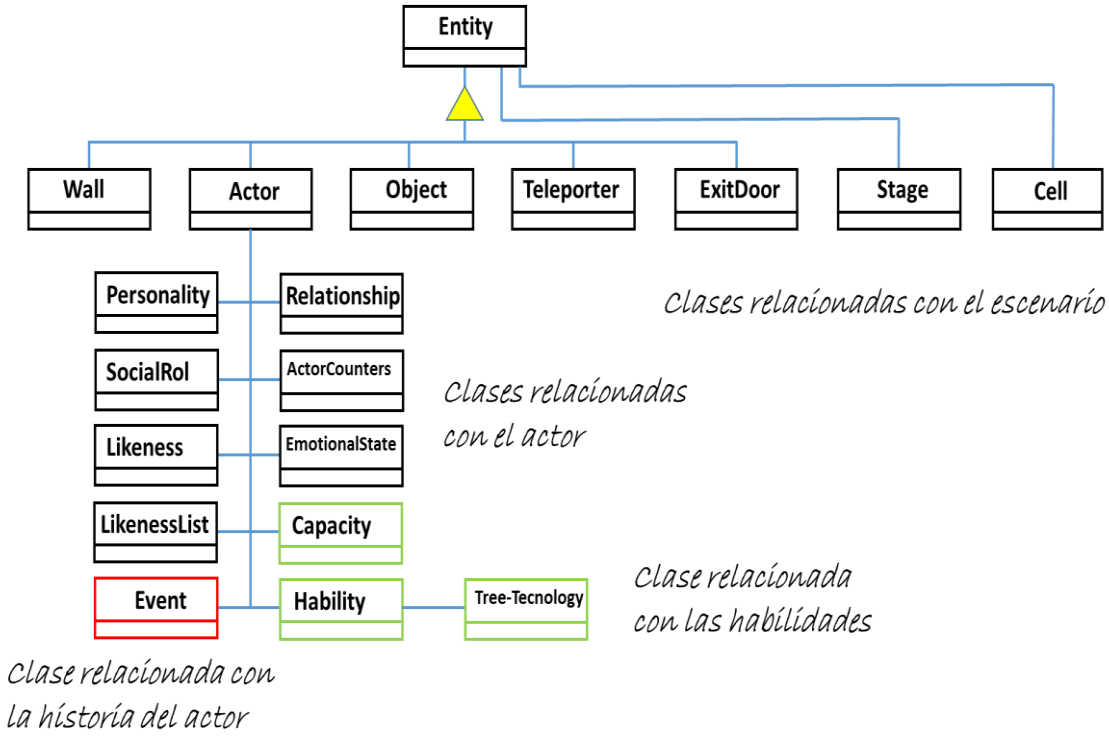

Ilustración 35: Clases Disponibles en AI-LIVE

La clase arriba indicada como "Event" es la clase creada para poder integrar la funcionalidad de la historia en el proyecto. Dentro de esta clase, se han tenido que crear especializaciones que hagan referencia a los distintos eventos de historia de los actores con sus atributos propios. A continuación mostramos el diagrama y explicaremos cada uno de ellos.

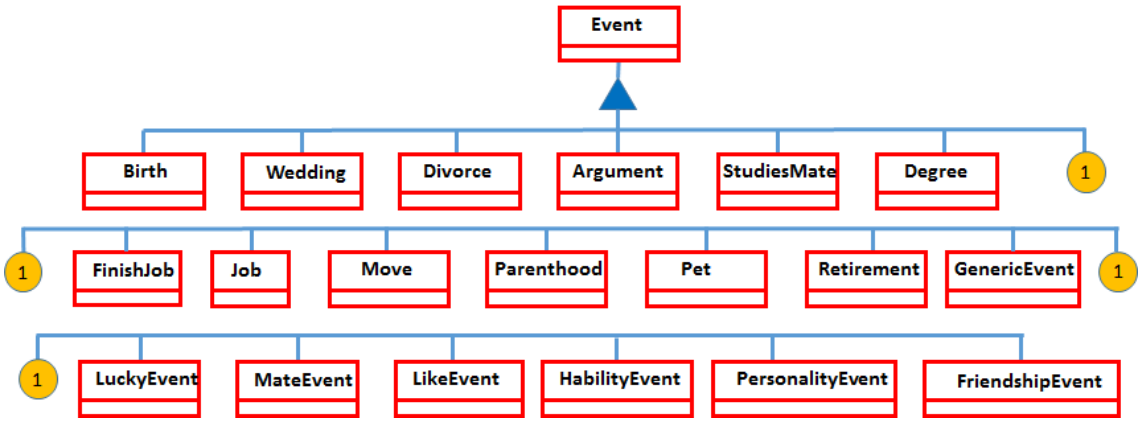

Ilustración 36: Clase y subclases de la historia de actor

El siguiente grupo de clases forman el conjunto de todos los eventos que van a estar disponibles puede realizar la historia de un actor en el universo AI-LIVE.

- Event: Es una generalización abstracta de todos los tipos de eventos de historia que existirá en el juego. Contiene los atributos comunes a todos ellos. Los atributos que componen esta clase son:
    - Id: campo de tipo INSTANCE-NAME utilizado para identificar al sim al que pertenece el evento. Hará referencia a la instancia del tipo Actor para asociarlo como evento al sim.
    - KindEvent: campo de tipo SYMBOL que indica el tipo de evento de historia.
    - Location: campo de tipo SYMBOL que recoge la localidad en donde tiene lugar el evento de la historia. Como valores disponibles estaría Madrid, Barcelona y Londres.
    - InitialDate: campo de tipo INSTANCE-NAME utilizado para identificar la fecha inicial del evento. Hace referencia a una nueva clase de nombre "Date" que contiene como atributos "day", "month" y "year".
    - FinalDate: campo de tipo INSTANCE-NAME utilizado para identificar la fecha inicial del evento. Hace referencia a una nueva clase de nombre "Date" que contiene como atributos "day", "month" y "year".
    - Genre: campo de tipo SYMBOL que recoge el tipo de evento en donde se podría englobar el evento de la historia. Como valores disponibles estaría Neutral, Childhood, Adventure, Sportive, Cultural, Social, Hability y Lucky.
    - Narratedstory: campo de tipo STRING que recoge la historia del evento en lenguaje legible.

- Birth: Especialización de la clase Event. Representa un evento de tipo nacimiento del sim. Los atributos propios que componen esta clase son:
    - Parents: Campo multi-valor de tipo INSTANCE-NAME que hacen referencia a las instancias de tipo Actor para asociarlos como padres del sim.

- Wedding: Especialización de la clase Event. Representa un evento de tipo boda del sim. Los atributos propios que componen esta clase son:
    - Partner: campo de tipo INSTANCE-NAME utilizado para identificar al sim con quien contrae matrimonio. Hará referencia a la instancia del tipo Actor para asociarlo como evento al sim.

- Divorce: Especialización de la clase Event. Representa un evento de tipo divorcio del sim. Los atributos propios que componen esta clase son:
    - Partner: campo de tipo INSTANCE-NAME utilizado para identificar al sim con quien rompe matrimonio. Hará referencia a la instancia del tipo Actor para asociarlo como evento al sim.

- Argument: Especialización de la clase Event. Representa un evento de tipo discusión del sim. Los atributos propios que componen esta clase son:
    - Partner: campo de tipo INSTANCE-NAME utilizado para identificar al sim con quien discute. Hará referencia a la instancia del tipo Actor para asociarlo como evento al sim.
    - Discussion: campo de tipo SYMBOL que recoge el asunto sobre el que discuten.

- StudiesMate: Especialización de la clase Event. Representa un evento de tipo compañero de estudios del sim. Los atributos propios que componen esta clase son:
    - Partner: Campo multi-valor de tipo INSTANCE-NAME que hacen referencia a las instancias de tipo Actor para asociarlos como compañeros de estudios del sim.

- FriendshipEvent: Especialización de la clase Event. Representa un evento de tipo evento de amistad que comparte el sim con otro sim. Los atributos propios que componen esta clase son:
    - Partner: Campo multi-valor de tipo INSTANCE-NAME que hacen referencia a las instancias de tipo Actor para asociarlos como compañeros del evento de amistad del sim.

  - Friendevent: Campo de tipo STRING que recoge el evento de amistad que comparte el sim y el resto de compañeros.
  - Maxpartnersavailable: Campo de tipo NUMBER que indica el máximo de compañeros que pueden compartir este evento.
  - Minoccurs: Campo de tipo NUMBER que indica el valor mínimo de eventos previos para que los actores se conozcan previamente entre sí.

- HabilityEvent: Especialización de la clase Event. Representa un evento de tipo evento de habilidad que puede compartir o no con otro sim. Los atributos propios que componen esta clase son:
  - Partner: Campo multi-valor de tipo INSTANCE-NAME que hacen referencia a las instancias de tipo Actor para asociarlos como compañeros del evento de habilidad del sim.
  - Habevent: Campo de tipo SYMBOL que indica el tipo de habilidad al que hace referencia este evento.
  - Subhabevent: Campo de tipo STRING que recoge el evento de habilidad que tiene el sim.
  - Maxpartnersavailable: Campo de tipo NUMBER que indica el máximo de compañeros que pueden compartir este evento.
  - Minoccurs: Campo de tipo NUMBER que indica el valor mínimo de eventos previos para que los actores se conozcan previamente entre sí.
  - Typeofhability: Campo de tipo SYMBOL que indica si el evento de habilidad es positivo o negativo.

- PersonalityEvent: Especialización de la clase Event. Representa un evento que ha realizado el sim y que tiene que ver con su personalidad. Suele hacer referencia a actos o conductas realizadas por el personaje en función de los cinco rasgos de personalidad que tiene (agreeableness, conscientousness, extraversion, neuroticism y openness) al cargar el perfil del actor. Este evento puede ser compartido o no con otro sim siempre y cuando tengan una personalidad parecida. Los atributos propios que componen esta clase son:
  - Partner: Campo multi-valor de tipo INSTANCE-NAME que hacen referencia a las instancias de tipo Actor para asociarlos como compañeros del evento de personalidad del sim.

  - Pevent: Campo de tipo STRING que recoge el evento de personalidad que tiene el sim.
  - Maxpartnersavailable: Campo de tipo NUMBER que indica el máximo de compañeros que pueden compartir este evento.
  - Minoccurs: Campo de tipo NUMBER que indica el valor mínimo de eventos previos para que los actores se conozcan previamente entre sí.

- Degree: Especialización de la clase Event. Representa un evento de tipo evento de estudios que ha realizado. Los atributos propios que componen esta clase son:
  - TypeOfDegree: Campo de tipo SYMBOL que indica el tipo de estudios que ha realizado.
  - NameOfDegree: Campo de tipo SYMBOL que recoge el título educativo, en sí, que ha estudiado.

- Job: Especialización de la clase Event. Representa un evento de tipo evento de trabajo que ha realizado. Los atributos propios que componen esta clase son:
  - Company: Campo de tipo SYMBOL que indica la compañía en la que trabaja.
  - KindOfJob: Campo de tipo SYMBOL que recoge los estudios requeridos para el trabajo.

- FinishJob: Especialización de la clase Event. Representa un evento de tipo evento de finalización de trabajo. Los atributos propios que componen esta clase son:
  - Job: Campo de tipo SYMBOL que indica la instancia de trabajo al que hace referencia.
  - Reason: Campo de tipo SYMBOL que recoge los motivos de finalizar el trabajo.

- Move: Especialización de la clase Event. Representa un evento de tipo evento de mudanza.

- Parenthood: Especialización de la clase Event. Representa un evento de tipo evento de paternidad/maternidad. Los atributos propios que componen esta clase son:
    - Partner: Campo de tipo INSTANCE-NAME que indica la instancia de tipo Actor del sim con quien tuvo el hijo/a.
    - Child: Campo de tipo INSTANCE-NAME que indica la instancia de tipo Actor del sim que actuaría como hijo/a.

- Pet: Especialización de la clase Event. Representa un evento de tipo evento de mascota. Los atributos propios que componen esta clase son:
    - NameOfPet: Campo de tipo SYMBOL que indica el nombre de la mascota.
    - Animal: Campo de tipo SYMBOL que recoge el tipo de mascota.

- Retirement: Especialización de la clase Event. Representa un evento de tipo evento de jubilación.

- GenericEvent: Especialización de la clase Event. Representa un evento de tipo evento genérico que puede compartir o no con otro sim. Los atributos propios que componen esta clase son:
    - Partner: Campo multi-valor de tipo INSTANCE-NAME que hacen referencia a las instancias de tipo Actor para asociarlos como compañeros del evento de tipo genérico del sim.
    - SEvent: Campo de tipo STRING que recoge el evento genérico que tiene el sim.

- LuckyEvent: Especialización de la clase Event. Representa un evento de tipo evento de azar. Los atributos propios que componen esta clase son:
    - LuEvent: Campo de tipo STRING que recoge el evento de azar que tiene el sim.
    - Typeofluck: Campo de tipo STRING que indica el tipo de azar que tiene sim.

- MateEvent: Especialización de la clase Event. Representa un evento de tipo compañero de trabajo del sim. Los atributos propios que componen esta clase son:
    - Partner: Campo multi-valor de tipo INSTANCE-NAME que hacen referencia a las instancias de tipo Actor para asociarlos como compañeros de trabajo del sim.
    - Company: Campo de tipo SYMBOL que indica la compañía en la que trabaja.

- LikeEvent: Especialización de la clase Event. Representa un evento de tipo evento de gustos que puede compartir o no con otro sim. Los atributos propios que componen esta clase son:
    - Partner: Campo multi-valor de tipo INSTANCE-NAME que hacen referencia a las instancias de tipo Actor para asociarlos como sims con los que comparte este evento al tener gustos vinculados.
    - Likevent: Campo de tipo STRING que indica el tipo de gusto al que hace referencia este evento.
    - Sublikevent: Campo de tipo STRING que recoge el lugar donde se realiza el tipo de gusto.

Además de estas clases necesarias para la funcionalidad, se han creado otras (no mostradas en diagrama al no aportar en el modelo de conocimiento) para preparar instancias que permitan tener una historia coherente y variada.

Es interesante indicar que, en este fichero se ha decidido añadir unas constantes, que se explicarán más adelante, para poder configurar la preferencia a la hora de crear eventos de un determinado tipo así como el umbral por el que dos o más personas son afines en personalidad. Sobre la preferencia para crear eventos de un tipo u otro funciona a modo de peso en la creación de historia por si se quisiera, antes de arrancar el juego, dar más importancia en buscar eventos de un tipo determinado entre varios (Personalidad, Amistad, Gustos, Azar, Genérico o Habilidad) de manera que, modificando estos valores, dos ejecuciones distintas con los mismos actores darán resultados distintos.

4.6 Desarrollo del servidor

El servidor está compuesto por los módulos Principal e Inteligencia Artificial y este último a su vez está dividido en varios ficheros tales como "server.clp", "hability_engine_server.clp" y "emotional_engine_server.clp" encargados de verificar que la acción del cliente en turno se puede ejecutar y, una vez hecho esto, actualiza el estado.

El servidor, al tratarse del núcleo de AI-LIVE y estar en continua comunicación con los distintos clientes, ha tenido que ser modificado y expandido para poder albergar la funcionalidad de este proyecto.

Para incorporar la nueva funcionalidad de Generación Automática de Historias se ha hecho un profundo análisis para determinar la forma de desarrollar la solución sin impactar en el rendimiento de la aplicación.

Para crear la historia del personaje, se ha creado un nuevo fichero "story_engine_server.clp", implementado en CLIPS, siendo una nueva parte del módulo de Inteligencia Artificial y que recoge las reglas y funciones propias de la generación de historias.

La historia del personaje, al ser una parte propia del sim, se ha decidido que esté contenida en un campo multi-valor, de nombre "Story", con las instancias de los eventos de la historia dentro de la instancia Actor, por lo que se ha creído conveniente que se genere en el momento de crear dicho personaje cuando entra en el universo AI-LIVE.

Esta acción se realiza en "server.clp", gracias a la regla "addClient", en donde se toman los valores del personaje cargado del perfil elegido y se le asignan al sim. En esa regla, después de haberle asignado las habilidades correspondientes, se indica mediante un hecho que es necesario crear la historia de ese personaje.

Una vez la regla "addClient" ha finalizado, las reglas relacionadas con la Generación Automática de Historias comienzan. Para poder introducir las reglas de la historia en ese momento, se han subido sus prioridades de manera que se ejecuten justo después de finalizar "addClient" y sólo se ejecutarán una vez por personaje ya que todas verifican que haya un hecho de creación de historia para ese personaje, como se ha indicado al final del párrafo anterior, y ese hecho se eliminará una vez se haya terminado de crear la historia.

La idea a la hora de generar la historia del personaje es que haya una serie de eventos particulares que le hayan ido sucediendo al sim desde que nace hasta la fecha siendo visto a grosso modo como una línea temporal con sucesos. Por eso, todo evento debe contener el identificador del sim al que pertenece (INSTANCE-NAME de Actor), fecha inicial y fecha final (INSTANCE-NAME de Date), la localidad en la que ha sucedido este evento así como otros campos de utilidad.

De esta manera, al ser una sucesión de eventos, el primer evento que ocurre es el nacimiento del sim y se calcula basándose en la edad del personaje cargada desde su perfil. En este primer evento de nacimiento, se indica la fecha, se determina el azar que tendrá en su vida el personaje y se selecciona el lugar de nacimiento pudiendo ser este Madrid, Barcelona o Londres. Se ha decidido indicar varias localidades cerradas para que el resto de eventos vayan en consonancia entre sí y los vínculos entre distintos personajes también tengan esto en cuenta. Adicionalmente, hay determinados eventos que ocurren en lugares propios de estas localidades para otorgar mayor coherencia y lógica a la historia en general. Para este ejemplo, el evento de nacimiento tendrá dos campos haciendo referencia a los padres que estarán inicializados a vacío. Esto es así ya que podemos dejar abierto la posibilidad de que haya otros sims (que entren en un futuro a la aplicación AI-LIVE) que sean sus padres y que, al generar sus historias, tendrán un evento relacionado con la paternidad de ese sim y hará que los campos inicializados a vacío en el evento de nacimiento se informen con la de los sims protagonistas de la paternidad.

Para cada regla, se tienen en cuenta todos los eventos pasados (de ahí que hagan falta funciones que se encarguen de resolver las limitaciones de reglas en un sistema de producción) y nos ayudará a obtener eventos lógicos que no desentonen en la historia.

Cada vez que se crea un evento, éste se inserta en el campo “Story” del Actor ordenado de manera cronológica y se cambia de fecha. El cambio de fecha se hace de forma aleatoria a partir de la última fecha usada, de manera que se va aumentando de fecha por cada evento y la historia terminará cuando lleguemos a una fecha mayor a la actual (2015).

Al haber una nueva fecha, y esta fecha no ser mayor a la actual, se procede a seguir con la creación de la historia. Basándose en la edad del actor en el momento de la fecha, la localidad, su personalidad, los gustos, su azar o sus habilidades se van generando de forma secuencial nuevos eventos para este personaje ya que se van ejecutando las distintas

reglas disponibles. Si la fecha del evento corresponde, por ejemplo, con la adolescencia del personaje nos podremos encontrar con eventos relacionados con sus estudios superiores o si el evento corresponde con la madurez podemos tener eventos relacionados con trabajar, con eventos del tipo boda o eventos en los que dejaba el trabajo actual debido a que se mudaba o le despedían.

Para poder ir creando eventos de cualquier tipo, debe haber instancias predefinidas que hagan referencia a los tipos de evento de forma que se puedan ir seleccionando según corresponda. Es decir, si a un personaje le gustan las mascotas, para que pueda haber un evento en el que diga que ese sim tuvo una mascota, que esa mascota era un perro y que se llamaba Doraemon es necesario que haya una instancia predefinida con tipos de mascota (perro, gato, pez..) y otra con nombres (Doraemon, Garfield..) para poder ser seleccionado.

Aumentando fechas una y otra vez hasta llegar a la fecha actual se habrá creado una serie de eventos que harán referencia a momentos del personaje que estarán disponibles en el Actor, se eliminará el hecho de creación de historia para ese sim permitiendo que el juego AI-LIVE siga con las demás funcionalidades y al personaje se le modificará un atributo “Storied” de su clase “Actor” que hace referencia a si se tiene la historia completada que, por defecto, se inicia a “NO” para pasar a “YES”.

En esta explicación, se ha contemplado que había un único personaje en AI-LIVE, en el caso de que hubiera más, al ir accediendo uno tras otro en el servidor, el primer personaje crearía su historia con eventos propios (al no haber más sims). Pero, a partir de ese momento, el siguiente personaje, a la hora de crear su historia, podría tener eventos compartidos con el primero ya que hay reglas en “story_engine_server.clp” que busca vínculos entre sims (gustos comunes, personalidades parejas..) de manera que se pueden crear eventos nuevos que compartan N personajes, actualizando y aumentando las historias de todos aquellos sims involucrados. En estos casos, el primer sim se podría decir que tiene su historia completada (“Storied = YES”) y al segundo que acaba de acceder se diría que su historia se le está creando por lo que aún no tendría historia completada (“Storied = NO”).

Es por eso que, cuantos más personajes haya en el universo AI-LIVE, más probable es que haya cantidad de eventos comunes entre todos y las historias sean más ricas y variadas.

Por último, cabe mencionar que también se ha desarrollado la posibilidad de poder priorizar entre unos tipos de historias u otras (historias que prioricen eventos de personalidad, habilidad, gustos, genéricos…) antes de arrancar la aplicación. Esto se ha hecho mediante el uso de variables constantes:

- defglobal ?*personality*: Para priorizar eventos de personalidad del sim.
- defglobal ?*friendship*: Para priorizar eventos de amistad entre sims.
- defglobal ?*likeness*: Para priorizar eventos de gustos del sim.
- defglobal ?*lucky*: Para priorizar eventos de azar del sim.
- defglobal ?*default*: Para priorizar eventos genéricos del sim.
- defglobal ?*hability*: Para priorizar eventos de habilidad del sim.

Estas constantes están presentes en las prioridades de cada regla de manera que al valor fijo de la prioridad que ya tiene cada regla se le suma la constante pertinente y estará por delante o detrás del resto dependiendo del valor de esta constante.

Además de estas constantes, se han creado otras constantes que ayudan a configurar la forma en la que se debe generar la historia. A continuación se indica algún de las más importantes:

- defglobal ?*ThresholdPersonality*: Es usada como margen de error a la hora de comparar la personalidad del personaje con las posibles instancias disponibles de personalidad o habilidad. Es un número decimal con un rango que va desde 0 hasta 1, cuanto más alto sea el margen más desvirtuados serán los eventos de personalidad o habilidad y cuanto más bajo más afines serán los eventos con la personalidad del actor.
- defglobal ?*MinBeforeWedding*: Es un valor entero que indica la cantidad de eventos en común que tienen que tener dos sims previamente a poder casarse. De esta manera, se intenta que haya eventos juntos antes de poder casarse.
- defglobal ?*MinBeforeWedding*: Es un valor entero que indica la cantidad de discusiones (eventos Argument) que tienen que tener dos sims casados entre sí para poder divorciarse.
- defglobal ?*limit*: Es un valor entero que indica la cantidad de eventos distintos que tiene que tener una historia por cada uno de los que se haga referencia en la función limitevent que se explicará más adelante. Se busca limitar la existencia de

historias con demasiados eventos de un tipo determinado porque alguna de las reglas tengan más prioridad que las otras.

En este apartado veremos los cambios y aportaciones incluidas en el servidor.

4.6.1 Aportes en el servidor

En "server.clp" y en "hability_engine_server.clp" no se han añadido reglas nuevas debido a que la parte del proyecto Generación Automática de Historias va englobado en su propio fichero y este proyecto está pensado, en líneas futuras, para interacción entre sims. Lo que se ha añadido a "server.clp" es el método para poder imprimir la historia del actor. El método es el siguiente:

- Método printStory: Este método se encarga de seleccionar la historia de cada actor con historia propia y guardarla en lenguaje "humanizado" localmente como un fichero <IdCliente>.storynarrated. Este método se llama en el consecuente de cada finalización de inserción de evento o de inserción de sim en algún evento que permita multiactores, para mantener actualizado la historia de cada actor.

En "emotional_engine_server.clp" sí se han añadido nuevas reglas, ya que se decidió añadir un par de funcionalidades a modo de muestra de correcto funcionamiento con la historia de los sims, que indicaremos a continuación.

- Regla TellStory: Permite a un agente comunicar a otro su historia. Para que esta acción pueda realizarse deben existir al menos dos agentes en el mismo escenario.

- Regla RememberHability: Regla que permite el intercambio de conocimiento mediante el recuerdo de un evento de historia entre dos Actores, siempre y cuando ambos Actores se encuentre en la misma habitación y haya un evento de tipo habilidad.

4.6.2 Cambios en el servidor

En el punto anterior indicamos que no se habían añadido nuevas reglas ni funciones a los ficheros "server.clp" y "hability_engine_server.clp" del módulo de Inteligencia Artificial debido a que no era necesario. Eso no significa que no se haya modificado alguna regla con el fin de que el servidor fuera capaz de entender el nuevo fichero "story_engine_server.clp" y poder invocarle.

De esta manera, se ha incluido en "server.clp" la necesidad de cargar el módulo de historias así como que se tengan en cuentan los cambios producidos en la historia del

cliente al realizar una jugada por si hubiera alguna actualización. A la hora de crear por primera vez un actor recién conectado, se indica que debe tener una historia y así lo indica gracias a un hecho insertado que se tendrá en cuenta en el motor de historias.

4.6.3 Aportes en "story_engine_server.clp"

El hecho de que se creara un motor propio de historias es debido a la necesidad de separar la funcionalidad del resto de funcionalidades de la parte del servidor. Este motor se ejecuta al crearse un actor, pero no es usado a posteriori por lo que no tiene lógica introducirlo en otros motores. Las reglas y funciones de este motor se ejecutan después de crear el personaje que ha ingresado a AI-LIVE gracias a que se ha priorizado sobre el resto de funcionalidades. Se han usado funciones y reglas para hacer este proyecto que se explican a continuación:.

4.6.3.1 Funciones en "story_engine_server.clp"

La lógica de crear una historia de un sim hace necesario que se tenga que recorrer los eventos ya creados del propio sim o historias de otros sims para que el evento a crear mantenga una sintonía dentro del conjunto de la historia global. Por esta razón, hay reglas que precisan de una confirmación en ese mismo momento de determinados estados en la fecha del evento a crear, teniendo que hacer llamadas a funciones para disponer de la última actualización posible. Las funciones creadas son:

- Función narrate: Está función es la encargada de traducir a lenguaje legible y entendible toda la información de un evento de historia. Siempre se va a ejecutar después de que una regla que implica creación/modificación de un evento de historia finalice.

  Si se crea una regla de un nuevo tipo de evento de historia, habrá que implementar su equivalencia en esta función para poder disponer de esa información.

- Función marriage: Está función devuelve la situación sentimental de los dos actores en una fecha en concreto, comprueba si están solteros, si están casados entre sí o casados con otros actores.

  Dependiendo de este valor se podrá crear eventos de matrimonio o de divorcio (siempre y cuando otras condiciones de las reglas sean satisfechas).

- Función workingnow: Está función devuelve la situación laboral del actor en la fecha indicada. Comprueba si ya está trabajando, si ha finalizado un trabajo o si

nunca ha trabajado.

- Función valid-studies: Está función valida la lógica de los estudios, de manera que no pueda haber un evento de estudios universitarios anterior a un evento de estudios de bachillerato (a menos que hubiera un evento de FP por ejemplo).

- Función location-date: Está función devuelve la localización que tiene el sim en la fecha indicada como parámetro de entrada. Será llamada en multitud de reglas ya que muchas condiciones dependen de la ubicación anterior del sim para poder crear el evento haciendo uso de unos datos u otros.

- Función insert-in-order: La ejecución de esta función dará como resultado la posición que debe ocupar, desde el punto de vista cronológico, el evento que se va a insertar en el multislot de la historia del sim.
  De esta forma, cuando tengamos que recorrer el multislot historia de sim siempre tendremos la certeza de estar ante un "array" ordenado.

- Función create-final-date: Esta función tiene como entradas una fecha inicial y dos valores numéricos a modo de mínimo y máximo de días. Como salida, devolverá una fecha final.
  La forma de crear esta fecha final es eligiendo un número aleatorio entre el valor ?min y el valor ?max y sumándolo a la fecha inicial. Teniendo en cuenta que cada mes tiene un número de días se ha tenido que estudiar todas las casuísticas posibles por lo que parece una función más densa de lo que realmente es.

- Función create-date: Esta función devuelve una fecha aleatoria dentro del año que se recibe como entrada. El resultado será usado como fecha inicial en cualquier evento del tipo historia del sim.

- Función mate: A partir de tres fechas, se va a comprobar si una de ellas está comprendida entre las otras dos. Esto nos servirá para certificar que dos sims han estudiado/trabajado juntos entre las fechas que han estudiado o trabajado.

- Función free-time-event: Teniendo en cuenta que un evento puede durar varios días, esta función comprueba antes de crear otro evento que la fecha inicial no interfiere entre otros eventos ya que podría hacer perder consistencia a la historia del sim.

- Función CountEventArgument: Esta función devuelve el número de eventos hasta la fecha indicada como parámetro de entrada en la que los dos actores han discutido previamente.

- Función CountEventPartner: Esta función devuelve el número de eventos hasta la fecha indicada como parámetro de entrada en la que los dos actores han compartido alguna historia sin importar el tipo de evento.

- Función not-ending-job: Esta función nos indica si el actor sigue trabajando o no en tipo de trabajo indicado como entrada. Ya que es posible que un actor tenga dos etapas en las que trabaje en la misma empresa pero en fechas distintas, es necesario revisar la historia del usuario de esta manera y no haciendo una condición simple puesto que perderíamos esa segunda etapa.

- Función notsame: Esta función nos indica si el actor está ya incluido en un evento de multiactores o no. Antes de insertar en eventos multiactor, habrá que comprobar que el sim satisfaga todas las condiciones y que no esté ya incluido puesto que entraríamos en un bucle de inserción entrando en una dinámica errónea.

- Función validpersonality: Esta función nos indica si un actor es compatible a nivel de personalidad con un evento de personalidad. Se tiene en cuenta los valores del evento de personalidad con la personalidad en sí del sim y respetando el umbral definido como variable global.
  De esta forma, si el umbral es amplio, el actor podrá acceder a más eventos de personalidad pero se perderá la calidad de la historia.

- Función countevents: Esta función devuelve el número de eventos de un determinado tipo que tiene el sim durante toda su historia.
- Función limitevent: Esta función comprueba que no haya más proporción de eventos de un mismo tipo con respecto al total de la historia. Se apoya en la constante ?*limit* y, si por ejemplo, el valor de ?*limit* es 3, no va a permitir que una historia tenga más de un tercio de eventos de un determinado tipo con respecto al total de eventos de la historia.
- Función nottogether: Función que no permite crear dos eventos de un mismo tipo seguidos (a menos que no se le invoque a la función). De esta manera, podremos tener eventos variados espaciados en el tiempo.
- Función validweather: Función que comprueba la coherencia entre tipos de eventos y las fechas de los mismos. Como ejemplo, no permite que haya un evento de "Práctica de ski en La Pinilla, Madrid" en Agosto.

4.6.3.2 Reglas en "story_engine_server.clp"

Con la finalidad de tener una historia compleja y variada que no fuera repetitiva ni escueta se han creado multitud de reglas, apoyado en una base de hechos potente para las historias. Las reglas son las siguientes:

- Regla HabilityEvent: Esta regla crea un evento de tipo habilidad para el sim cuya historia está siendo creada. Se apoya en los factores de la responsabilidad y apertura a nuevas iniciativas de la personalidad y habilidades iniciales con las que se crea el personaje cuando se incorpora al universo AI-LIVE para crear un evento de este tipo. Este evento tendrá un atributo que indicará el máximo de personajes que podrán incorporarse en un futuro al mismo pero en esta regla no se añadirán actores puesto que para eso habrá una regla que irá añadiéndolos según satisfagan las condiciones.

- Regla InsertInHability: Regla que permite crear eventos de tipo habilidad con multi-personajes. Busca eventos de tipo habilidad, ya creados, en las historias de otros personajes del universo AI-LIVE para crear un evento nuevo en su historia personal con la información y actores del evento. Para los eventos de los distintos actores relacionados se deberá actualizar la información del nuevo personaje a incluir, pero esto lo llevará a cabo otra acción que se explicará más adelante.

Como esta regla busca eventos ya creados de otras historias de personajes, para poder acceder a ellos es preciso que se verifiquen algunas variables que debe tener en común el sim con los demás personajes del evento tales como personalidad parecida, residir en esa fecha en la misma localidad o disponer de la misma habilidad que la del tipo de evento así como haber tenido algún evento en común en el pasado con los distintos personajes.

- Regla PersonalityEvent: Esta regla crea un evento de tipo personalidad para el sim cuya historia está siendo creada. Se apoya en los 5 factores de la personalidad (Extraversión, Apertura al cambio, Responsabilidad, Amabilidad e Inestabilidad Emocional) del personaje que tiene cuando se incorpora al universo AI-LIVE para crear un evento de este tipo. Este evento tendrá un atributo que indicará el máximo de personajes que podrán incorporarse en un futuro al mismo pero en esta regla no se añadirán actores puesto que para eso habrá una regla que irá añadiéndolos según satisfagan las condiciones. Este tipo de eventos hacen referencia a acciones propias de la personalidad de un personaje dependiendo de si es introvertido o no, si es amable o si es responsable.

- Regla InsertInPersonality: Regla que permite crear eventos de tipo personalidad con multi-personajes. Busca eventos de tipo personalidad, ya creados, en las historias de otros personajes del universo AI-LIVE para crear un evento nuevo en su historia personal con la información y actores del evento. Para los eventos de los distintos actores relacionados se deberá actualizar la información del nuevo personaje a incluir, pero esto lo llevará a cabo otra acción que se explicará más adelante. Como esta regla busca eventos ya creados de otras historias de personajes, para poder acceder a ellos es preciso que se verifiquen algunas variables que debe tener en común el sim con los demás personajes del evento tales como personalidad parecida o residir en esa fecha en la misma localidad así como haber tenido algún evento en común en el pasado con los distintos personajes.

- Regla GenEvent: Esta regla crea un evento de tipo genérico para el sim cuya historia está siendo creada. Se trata de una regla que genera eventos menos específicos o exclusivos que otros como pueden ser los eventos de personalidad o

habilidad. Este evento hace referencia a acciones cotidianas que puede hacer una persona tales como salir a dar un paseo o visitar a los abuelos. Tiene en cuenta la edad del sim en ese momento para determinar el tipo de acción a realizar. No añade actores al evento puesto que para eso habrá una regla que vaya añadiéndolos según satisfagan las condiciones.

- Regla InsertInGeneric: Regla que permite crear eventos de tipo genérico con multi-personajes. Busca eventos de tipo genérico, ya creados, en las historias de otros personajes del universo AI-LIVE para crear un evento nuevo en su historia personal con la información y actores del evento. Para los eventos de los distintos actores relacionados se deberá actualizar la información del nuevo personaje a incluir, pero esto lo llevará a cabo otra acción que se explicará más adelante.

- Regla Friendship: Esta regla crea un evento de tipo amistad para el sim cuya historia está siendo creada y otro personaje que está ya en el universo AI-LIVE. Esta regla, a diferencia de las reglas de personalidad, habilidad o genérica, no crea un evento inicial con un único personaje y, posteriormente, otra regla se encarga de ir insertando personajes sino que este evento siempre parte de dos personajes. Para que se puedan crear este tipo de eventos se revisan los eventos pasados para comprobar en cuántos eventos ha coincidido el sim, cuya historia está siendo creada, con cada personaje del universo AI-LIVE y aquel que tenga un mínimo de ocurrencias con el primero se creará un evento que tenga relación con uno de los gustos del personaje aunque al otro no le guste pero lo realiza como muestra de amistad. No añade más actores al evento puesto que para eso habrá una regla que vaya añadiéndolos según satisfagan las condiciones.

- Regla InsertInFriendship: Esta regla busca un evento de tipo amistad de dos sims presentes en el universo AI-LIVE y se incluye el sim cuya historia está siendo creada actualizando los eventos ya creados. Como cada sim debe tener su propia historia, se crea un evento igual al que ya están creados para el sim actual.

- Regla InsertingPartner: Esta regla añade a un evento ya creado la información de un nuevo sim que se ha añadido por ser regla multi-personaje tal como

InsertInGeneric o InsertInFriendship. Se ejecuta después de las reglas recién mencionadas para mantener estos eventos consistentes con los nuevos actores añadidos. Para identificar aquellos eventos que precisan ser actualizados con la información de nuevos personajes es preciso que anteriormente se haya ejecutado la regla LookingInsertsEvents que explicaremos a continuación.

- Regla LookingInsertsEvents: Esta regla busca eventos de tipo multi-personaje en cada historia de los personajes del universo AI-LIVE en donde haya distinto número de actores para indicar que se tiene que actualizar mediante la regla InsertingPartner.

- Regla JobMate: Esta regla crea un evento para el personaje cuya historia está siendo creada y otro personaje que ya está presente en AI-LIVE. Se trata de un evento que indica cuándo comenzaron a ser amigos dos compañeros de trabajo. Para ello, es necesario que ambos personajes estén trabajando en la misma compañía en la fecha a crear el evento y deben tener en común un gusto por algo.

- Regla LEvent: Esta regla crea un evento de tipo gusto para el personaje cuya historia está siendo creada y otro personaje que ya está presente en AI-LIVE. Se trata de un evento que indica una acción que realizaron estos dos personajes con un gusto en común. La acción que realizan en el evento hace referencia a ese gusto en común.

- Regla LuckEvent: Esta regla crea eventos para el personaje cuya historia está siendo creada y la acción que indica el evento hace referencia al tipo de suerte que tenga. Esta suerte está asignada al nacer mediante la regla Birth y le acompañará en toda su historia. De esta manera, si un personaje tiene buena suerte en la vida, los eventos de este tipo serán positivos.

- Regla Argue: Esta regla crea un evento de tipo discusión para el personaje cuya historia está siendo creada y otro personaje que ya está presente en AI-LIVE. Se trata de un evento que hace referencia a un enfrentamiento entre estos dos personajes. Pueden discutir sobre distintos motivos eligiéndose al azar uno de

ellos. Este evento de discusión puede hacer crear eventos entre estos personajes posteriormente a modo de repercusión negativa tales como divorcios o despidos de trabajo.

- Regla Fired: Esta regla se ejecuta cuando hay un evento previo de discusión (regla Argue) entre compañeros de trabajo y uno de ellos es el jefe del otro. Esta regla añade a la base de conocimientos del sistema la necesidad de crear un evento de finalización de trabajo (Regla FinishJob) por despido y modifica la fecha de finalización en el evento de trabajo del personaje en cuestión.

- Regla LeavingJob: Esta regla se ejecuta cuando dos eventos previos, uno de cambio de domicilio (regla Move) a otra localidad distinta a la actual y otro de trabajo (regla Job) cuya fecha de finalización aun no este informado indicando que sigue trabajando ahí. Esta regla añade a la base de conocimientos del sistema la necesidad de crear un evento de finalización de trabajo (Regla FinishJob) de tipo baja voluntaria de trabajo y obliga a modificar la fecha de finalización en el evento de trabajo del personaje en cuestión.

- Regla Parenthood: Esta regla crea un evento de paternidad-maternidad entre dos sims, el personaje cuya historia está siendo creada y otro personaje que ya está presente en AI-LIVE. Es un evento muy exclusivo ya que requiere que se cumplan muchas condiciones para que se pueda generar. Ambos personajes deben estar casados (regla Wedding), siendo de sexo opuesto y debe haber otro sim que, por edad entre otras condiciones, pueda encajar. Cuando se genera el evento de paternidad también se modifica el evento de nacimiento del hijo con la información de los padres.

- Regla Job: Esta regla crea un evento de trabajo del personaje cuya historia está siendo creada. El trabajo que se le asigne al personaje estará acorde a los estudios y gustos del sim de forma que si hay una persona ha cursado estudios universitarios podrá optar a un puesto de trabajo mejor que uno con estudios básicos.

- Regla Retire: Esta regla crea un evento sobre la jubilación del personaje cuya historia está siendo creada. Es necesario que el personaje tenga más de 67 años y esté trabajando en el momento de creación del evento. Esta regla añade a la base de conocimientos del sistema la necesidad de crear un evento de finalización de trabajo (Regla FinishJob) por jubilación y modifica la fecha de finalización en el evento de trabajo del personaje en cuestión.

- Regla FinishJob: Esta regla crea un evento haciendo referencia al motivo por el que finaliza el trabajo que tenía en curso el personaje cuya historia está siendo creada. Es necesario que se haya disparado previamente una de las reglas (Fired, Retire o LeavingJob) que indican la necesidad de finalizar el trabajo debido a distintos motivos (despido, jubilación, cambio de localidad de residencia).

- Regla HighStudies: Esta regla crea un evento sobre los estudios superiores que ha realizado el personaje cuya historia está siendo creada. Puede estudiar tanto Bachelor (Bachiller), Vocational Training (Grado medio) o University (Universidad) y se tienen en cuenta los gustos del personaje para seleccionar el tipo de estudios a realizar. Dependiendo de la edad del personaje en el momento del evento también se tendrá en cuenta el tipo de estudios (con 16 años no puedes empezar la universidad pero sí el bachiller por ejemplo).

- Regla Divorce: Esta regla crea un evento de divorcio entre dos sims, el personaje cuya historia está siendo creada y otro personaje que ya está presente en AI-LIVE. Es un evento muy exclusivo ya que requiere que se cumplan muchas condiciones para que se pueda generar. Ambos personajes deben estar casados (regla Wedding) y haber tenido hasta la fecha un mínimo de eventos de discusión (regla Argue) entre sí que haya propiciado la ruptura.

- Regla Wedding: Esta regla crea un evento de boda entre dos sims, el personaje cuya historia está siendo creada y otro personaje que ya está presente en AI-LIVE. Es un evento muy exclusivo ya que requiere que se cumplan muchas condiciones para que se pueda generar. Ambos personajes no pueden estar casados, pero sí pueden estar divorciados y no sólo solteros, y deben haberse visto hasta la fecha

un mínimo de veces entre sí que haya propiciado la boda.

- Regla Move: Esta regla crea un evento referente a un cambio de domicilio que ha realizado el personaje cuya historia está siendo creada. El cambio de domicilio puede ser en la misma localidad u otra. Si es un cambio de localidad, este evento puede abrir nuevos eventos en el futuro relativos al trabajo actual que haga abandonar el trabajo por este cambio de localidad.

- Regla Pet: Esta regla crea un evento referente a la mascota que ha tenido el personaje cuya historia está siendo creada. Es un tipo de eventos exclusivo para los amantes de las mascotas y así se tendrá en cuenta a la hora de crear el evento ya que es necesario que al personaje le gusten las mascotas.

- Regla StudiesMate: Esta regla crea un evento para el personaje cuya historia está siendo creada y otro personaje que ya está presente en AI-LIVE. Se trata de un evento que indica cuándo comenzaron a ser amigos dos compañeros de estudios. Para ello, es necesario que ambos personajes estén estudiando lo mismo en la misma localidad en la fecha a crear el evento y deben tener en común un gusto por algo.

- Regla School: Esta regla crea un evento sobre los estudios obligatorios que ha realizado el personaje cuya historia está siendo creada. En este caso el evento engloba la acción de los estudios hasta la finalización de los estudios primarios y secundarios. Este evento siempre ocurre para cualquier historia de personajes en AI-LIVE y permitirá abrir nuevos vías para trabajos o estudios superiores.

- Regla Birth: Esta regla crea el primer evento de cualquier personaje cuya historia está siendo creada. En esta regla se determina la localidad en la que nació, muy importante puesto que el resto de eventos de la historia tendrá que ver con esa localidad a menos que se cambie de localidad (regla Move), la suerte que esta persona tendrá durante toda su vida, y permitiendo que pueda presentar eventos de azar (regla LuckEvent) del personaje. Este evento contiene dos atributos haciendo referencia a los nombres de los padres de este sim que en el momento

de crearse estarán sin informar a la espera de que haya eventos de tipo paternidad (regla Parenthood) que actualice esta información. Se ha creído oportuno dejarlo sin informar, en lugar de poner nombres genéricos, ya que nos resulta más interesante que los padres sean personajes del universo AI-LIVE y dejar abierta la puerta a futuras evoluciones sobre forma de actuar entre parientes.

- Regla NonRulesAvailable: Esta regla se dispara si no hay una regla que crea un evento. Está pensado para ocasiones en las que no haya ninguna regla que genere eventos que cumpla las condiciones en una determinada fecha evitando que se quede el proceso de generación de historia en un bucle que no pueda salir. Esta regla modificará la fecha para que se vuelvan a calcular las posibles reglas que satisfagan las condiciones para generar eventos de historia.

- Regla insertInstance: Después de que se genere un evento de historia del personaje, se indica la necesidad de que ese evento se incorpore a la historia del mismo. Esta historia es una lista de eventos y la regla se encargará de ingresarlo en la lista en la posición que se haya indicado previamente gracias a la función "insert-in-order".

- Regla newchangeYear: Después de cada generación de evento de historia, es necesario que se genere un nuevo registro de fecha que hará referencia al próximo evento. Esta regla es la encargada de cambiar la fecha.

- Regla endStory: Esta regla finaliza la creación de la historia del sim. Se ejecuta cuando la fecha disponible para crear nuevos eventos es mayor a la fecha actual por lo que no tiene sentido seguir creando eventos al encontrarnos en la actualidad. Finaliza la necesidad de crear la historia al personaje puesto que lo acaba de hacer, imprimo las instancias de la historia en el estado del actor e invoco una función ("printStory") que escribe en un fichero la historia en un lenguaje que sea más sencillo de entender para cualquier persona.

4.7 Desarrollo en los clientes CLIPS

En los clientes CLIPS de tipo manual e Inteligencia Artificial se han tenido que añadir nuevos métodos y reglas para poder adecuar y explotar la funcionalidad creada con el servidor de historias de clientes.

Con el fin de mostrar algún ejemplo del potencial que pueden ofrecer las historias de los personajes del universo AI-LIVE se han creado dos acciones bastantes sencillas que se explicarán a continuación.

Llegados a este punto, en donde los actores ya han sido creados, las historias de cada personaje ya están generadas y pueden ser usadas en, por ejemplo, acciones que tengan que ver con la interacción entre sims.

La acción más básica, de las dos que se han creado en relación a la historia de personajes, es la que un sim le cuenta a otro su historia personal provocando cambios en los estados emocionales de ambos sims.

La segunda acción es más compleja y es aquélla en la que un sim le cuenta a otro sim (con menos habilidad en un ámbito que el primero) un evento de su historia personal que guarda relación con la de la habilidad en cuestión. En este caso la acción va a hacer que el sim oyente pueda aprender y mejorar su habilidad en ese campo.

Una vez seleccionada una de estas acciones, se enviaría mediante los métodos correspondientes (y que veremos más abajo) la información al servidor con el fin de que se puedan validar, ejecutar y actualizar el escenario.

A continuación, procedemos a indicar los métodos y reglas que se han implementado.

4.7.1 Métodos en “client.clp”

Hay dos nuevos métodos creados en la parte cliente que hace referencia a la acción a enviar al servidor. La única labor de los métodos de client.clp es escribir la acción que va a realizar en el fichero current.action. Debe existir un método que realice esta operación por cada acción distinta que pueda enviarse al servidor. De esta forma, encontramos:

- Método printRememberStoryClientAction: Envía la acción a realizar al servidor con la información que tiene la acción relativa a la regla RememberHabilityEvent.

- Método printVerbalStoryClientAction: Envía la acción a realizar al servidor con la información que tiene la acción relativa a la regla TellStory.

4.7.2 Reglas en "client.clp"

En cuanto a las reglas nuevas se refiere, se han creado dos nuevas reglas para poder demostrar algo del potencial que puede ofrecer disponer de una historia por actor. Así pues, encontramos:

- Regla RememberHabilityEvent: Regla que permite transmitir conocimiento entre dos actores a través de un evento del pasado. El Actor que presente el nivel más alto de conocimiento transmitirá dicha habilidad al otro Actor que aprenderá el conocimiento transmitido.

- Regla TellStory: Regla que permite contar la historia de un sim a otro que estén situados uno al lado del otro.

5. Pruebas y resultados

En este apartado se describen las pruebas que se han efectuado con el fin de verificar las funcionalidades implementadas en el proyecto de Generación Automática de Historias y analizar las salidas obtenidas de las ejecuciones.

Tal y como hemos indicado a lo largo de esta memoria, la creación de historias dependen de muchos factores, tanto del actor como del resto de actores que ya están en el universo AI-LIVE con sus historias ya creadas.

Para poder probar la correcta funcionalidad será necesario realizar varias pruebas cambiando las características de los actores (edad, personalidad, habilidades, gustos, azar, sexo…) y comprobar que los resultados tengan lógica con la entrada inicial.

Por último, probaremos la funcionalidad de interacción entre sims en la que un actor recuerda un evento de su historia relacionado con una habilidad a otro y este último es capaz de adquirir conocimiento de esa habilidad.

- **Prueba 1: Ejecución de un único actor en el universo AI-LIVE**

  Con esta prueba se pretenden dos cosas, una, confirmar que la creación de esta funcionalidad no impacta en el normal funcionamiento del resto de funcionalidades ya implantada anteriormente y, dos, observar que, al tener únicamente un actor, la historia que se creará se basará en eventos más genéricos basados en la edad, su habilidad o personalidad.

  Se ha ejecutado un solo actor:

  ./run.sh -s -c mike

  Vamos a ver algunas características del actor que va a ingresar, de nombre Mike:

```
([DEFAULT_ACTOR] of AddClient
(name_ "Mike")
(gender M)
(agreeableness 0.2)
(conscientousness 0.7)
(extraversion 0.1)
(neuroticism 0.8)
(openness 0.2)
(age 75)
(status 0)
(sex 0)
(theoretical 4)
(practical 2)
(modo Hability)
(habilities Informatics Logic Physical)
(level_hability 6 5 6)
```

Mike tendría 75 años, varón, poco agradable, constante, una persona introvertida, neurótico y poco abierto a cambios. Le gustan los ordenadores, navegar por internet, las películas o las matemáticas entre otras cosas.

Así pues, se procede a la ejecución. Como no hay más personajes en el universo AI-LIVE, la historia carecerá de eventos más complejos con actores reales en donde puedan compartir actividades teniendo en cuenta su personalidad, sus gustos o habilidades.

Tal y como está diseñado, hay determinados eventos que siempre deben ejecutarse para que tenga cierta lógica la historia. La primera de todas, es el nacimiento, es necesario que haya un evento relativo a cuándo y dónde nació. Así pues, se ejecutará la "regla Birth" asignando una fecha de nacimiento partiendo de la edad de Mike y el lugar de Nacimiento (selección de Madrid, Barcelona o Londres) y la suerte que tendrá Mike a lo largo de la vida.

Una vez creado el nacimiento, la lógica de la historia dice que toda persona habrá tenido estudios primarios y secundarios al ser obligatorios, cosa que deja constancia el evento Degree. Posteriormente, durante la generación de la historia, el personaje podrá mostrar nuevos eventos de tipo Degree (bachillerato, grado medio, universidad) teniendo en cuenta los gustos del sim.

Al disponer ya de estos dos eventos iniciales, el resto de eventos irán en función de la localidad en la que haya nacido o de la localidad a la que se haya mudado más tarde, de los estudios, gustos, habilidades o personalidad así como de los actores que haya a su alrededor.

En este caso, al ser un único actor, la historia se basa en eventos genéricos que tienen que ver con la edad en el momento de generar el evento, como jugar al escondite cuando Mike era joven o viajar con el imserso tras estar jubilado, también encontramos eventos de estudios de grado medio (Técnico Informático ya que entre sus gustos están "Computers" (Ordenadores) e "Internet") e incluso un trabajo relacionado con sus estudios.

Se ven eventos que hacen referencia a la buena suerte que tiene el usuario durante su vida, como encontrarse una maleta con dinero o conocer a un famoso en la calle,

tiene eventos que tienen que ver con su personalidad tales como ir al cine con sus amigos (se trata de una persona introvertida por lo que no tendría lógica que saliera con gente desconocida), fugarse de casa o irse antes de una fiesta por tener una discusión (recordemos que el personaje es una persona desagradable e introvertida).

Además, el hecho de estar trabajando hasta la fecha de la jubilación hace que aparezca un evento haciendo referencia a su jubilación y el evento en el que deja de trabajar debido a que ha llegado a la edad de jubilarse. Estos eventos no se ejecutarían de no tener anteriormente un evento activo de trabajo y haber llegado a la edad propia de jubilación.

La traza que se genera después de la generación de la historia nos muestra lo siguiente:

---STORY OF [CLIPS_ACTOR_TZMMQAZ936]---

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind Birth in London at 18/4/1940.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind Degree in London from 1/9/1944 until 14/6/1956. CLIPS_ACTOR_TZMMQAZ936 studied Primary_Secondary_Education at School.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind GenericEvent in London at 21/11/1948. The event was play hide-and-seek.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind Degree in London from 15/9/1956 until 8/6/1958. CLIPS_ACTOR_TZMMQAZ936 studied Technician at Vocational_Training.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind LuckyEvent in London at 19/8/1958. This was a Good luck event and it was winning a tv contest.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind PersonalityEvent in London at 26/2/1961. This event was go to the cinema with best friends.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind Job in London from 13/9/1965 until 3/6/2007. CLIPS_ACTOR_TZMMQAZ936 worked as a HelpDesk at Marsans.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind Move in London at 3/8/1972.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind LuckyEvent in London at 19/10/1976. This was a Good luck event and it was find a suitcase full of money.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind GenericEvent in London at 1/9/1978. The event was buy a car.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind PersonalityEvent in London at 21/12/1985. This event was run away from home.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind GenericEvent in London at 19/11/1993. The event was get driving license.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind PersonalityEvent in London at 21/11/2001. This event was leave early from a party due to an argument.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind GenericEvent in London at 15/12/2003. The event was go on strike.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind Retirement in London at 3/6/2007.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind FinishJob in London at 3/6/2007. CLIPS_ACTOR_TZMMQAZ936 stopped working as a HelpDesk at Marsans due to Retirement.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind GenericEvent in London at 4/6/2008. The event was go to a spa with the imserso.

Sim CLIPS_ACTOR_TZMMQAZ936 had an event of kind LuckyEvent in London at 21/4/2014. This was a Good luck event and it was meet a famous at the street.

---END OF STORY OF [CLIPS_ACTOR_TZMMQAZ936]---

De esta manera, podemos afirmar que la funcionalidad está correcta, no afecta al resto de funcionalidades del proyecto y, al ser un único actor, la historia se basa en eventos de su propia persona sin interacción de terceros.

- **Prueba 2: Ejecución de varios actores en el universo AI-LIVE con pocas compatibilidades entre sí y dando más peso a los eventos de AMISTAD.**

Esta prueba sirve para demostrar que, si aumentamos el pesaje de una determinada constante de prioridad, la historia a crear intentará buscar por delante del resto de tipologías aquellas que tengan mayor peso o prioridad. En este caso, vamos a probar dando más peso a la amistad (?*friendship*). Esto no quiere decir que todos los eventos sean de AMISTAD sino que, en caso de igualdad de elección, se elegirá este en primer lugar.

Además, se va a mostrar como una historia inicial puede ir aumentando a medida que otros actores van incorporándose y se crean eventos entre sí.

La regla que valora crear eventos de amistad crea actividades que son más íntimas y exclusivas que un evento genérico. Es decir, un evento genérico puede ser irse de compras revisando la localidad o la edad que tiene pero en el caso de eventos de

amistad, es necesario que las personas que vayan a compartir ese evento con el actor hayan tenido eventos en común anteriormente y que, al menos, a uno de ellos les guste la actividad a realizar. Se supone que muchas veces hacemos alguna actividad tan solo porque al amigo con quien vas a realizarla le gusta.

Se modifica la constante que gestiona el pesaje sobre eventos:

*(defglobal ?*personality* = 0)*

(defglobal ?*friendship* = 5)

*(defglobal ?*likeness* = 0)*

*(defglobal ?*lucky* = 0)*

*(defglobal ?*default* = 0)*

*(defglobal ?*hability* = 0)*

Se modifica la constante que gestiona el margen que modifica las personalidades de los actores para ampliar el abanico de posibilidades para eventos de personalidad y habilidad (aunque la calidad en la relación entre la personalidad y el evento será menor):

(defglobal ?*thresholdPersonality* = 0.6)

Se han ejecutado dos actores:

./run.sh -s -c sally -c charlie

En este caso, se han ejecutado 2 actores, distintos entre sí y con un único gusto en común entre ellos por lo que limitará la creación de eventos a compartir pero, en caso de que ocurra alguno, habilitará la posibilidad de crear algún evento de amistad al tener un peso mayor.

Aunque se van a mostrar cada una de las trazas de cada actor como en la prueba 1 con los eventos que ha tenido, se va a intentar explicar gráficamente la historia de cada uno de los actores y cómo han ido evolucionando sus propias historias al añadirse nuevos actores y eventos en común por cada nuevo actor.

Características de Sally:

```
([DEFAULT_ACTOR] of AddClient
(name_ "Sally")
(gender F)
(agreeableness 0.4)
(conscientousness 0.6)
(extraversion 0.4)
(neuroticism 0.5)
(openness 0.3)
(age 63)
(status 0)
(sex 1)
(theoretical 4)
(practical 2)
(modo Hability)
(habilities Logic Maths)
(level_hability 6 5)
(generLike Walking Painting Clothes Animals Mountain Books Photo)
```

Características de Charlie:

```
([DEFAULT_ACTOR] of AddClient
(name_ "Charlie")
(gender M)
(agreeableness 0.9)
(conscientousness 0.5)
(extraversion 0.6)
(neuroticism 0.4)
(openness 1)
(age 48)
(status 0)
(sex 0)
(theoretical 4)
(practical 2)
(modo Hability)
(habilities Cooking Sport)
(level_hability 6 5)
(generLike Action Films BallSports Snow Mountain Food)
```

- Primera ejecución. Jugadores en AI-LIVE: SALLY

El primer personaje es Sally, vamos a mostrar su historia sobre una línea de tiempo desde 1952 (ya que el personaje tiene 63 años) y sobre ella los distintos eventos, con las particularidades de cada evento, la localidad y el año en el que sucede:

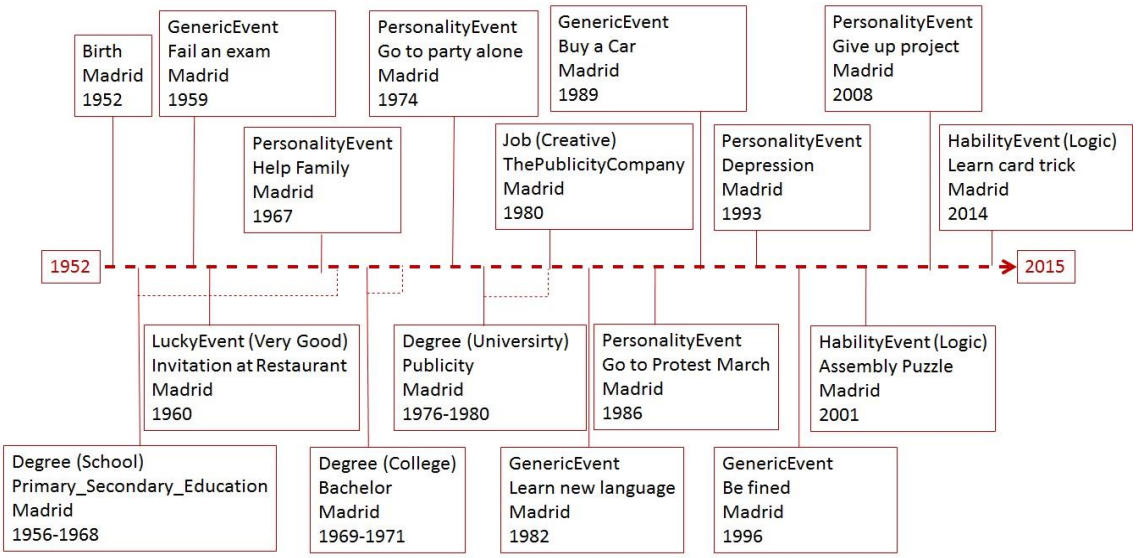

Ilustración 37: Historia Sally. 1 ejecución

Si estudiamos los eventos ocurridos vemos que Sally ha vivido siempre en Madrid. Encontramos eventos genéricos que tienen que ver con su juventud (suspender un examen en el colegio), eventos relacionados con sus estudios (Estudios secundarios, bachillerato así como universidad) o eventos que hacen referencia a su personalidad, habilidad o azar.

El haber estudiado la carrera de Publicidad en la universidad y gustarle la fotografía (Photo), le abre la puerta a trabajar como creativo en una empresa del sector de marketing.

Si miramos la personalidad de Sally nos encontramos con que es una persona con valores normales, pero el hecho de que haya un margen elevado (?thresholdpersonality* = 0.6), para eventos de habilidad y personalidad, hace que pueda abarcar cualquier tipo de evento de personalidad, desde eventos para personas más abiertas (ir a una fiesta sola) a eventos para personas más paranoicas (estar deprimida) haciendo que los eventos de personalidad sean poco certeros.

También se encuentran eventos de tipo habilidad. Sally tiene como habilidades la lógica y las matemáticas con una constancia de 0.6 por lo que se podría decir que es una persona concienzuda en aquello que se propone y al hablar de habilidad podríamos esperar que consiguiera eventos de habilidad positivos. El que haya un margen de personalidad amplio (?*thresholdPersonality* = 0.6) podría haber hecho

encontrarnos con eventos de habilidad malos o buenos pero, en este caso, han resultado ser buenos (crear un puzle y aprender nuevo truco de cartas).

Como resumen a esta primera ejecución, al no tener más actores con los que poder tener eventos más exclusivos, se han creado eventos genéricos basados en su edad y localización en cada momento de la historia, eventos relativos a su personalidad, estudios, a su habilidad o a su azar.

- Segunda ejecución. Jugadores en AI-LIVE: SALLY y CHARLIE

Tras Sally, Charlie entra en AI-LIVE y es necesario crear su historia. Vamos a mostrar su historia sobre una línea de tiempo desde 1967 (el actor tiene 48 años) y sobre la misma los distintos eventos que le ocurren en esa primera ejecución de historia:

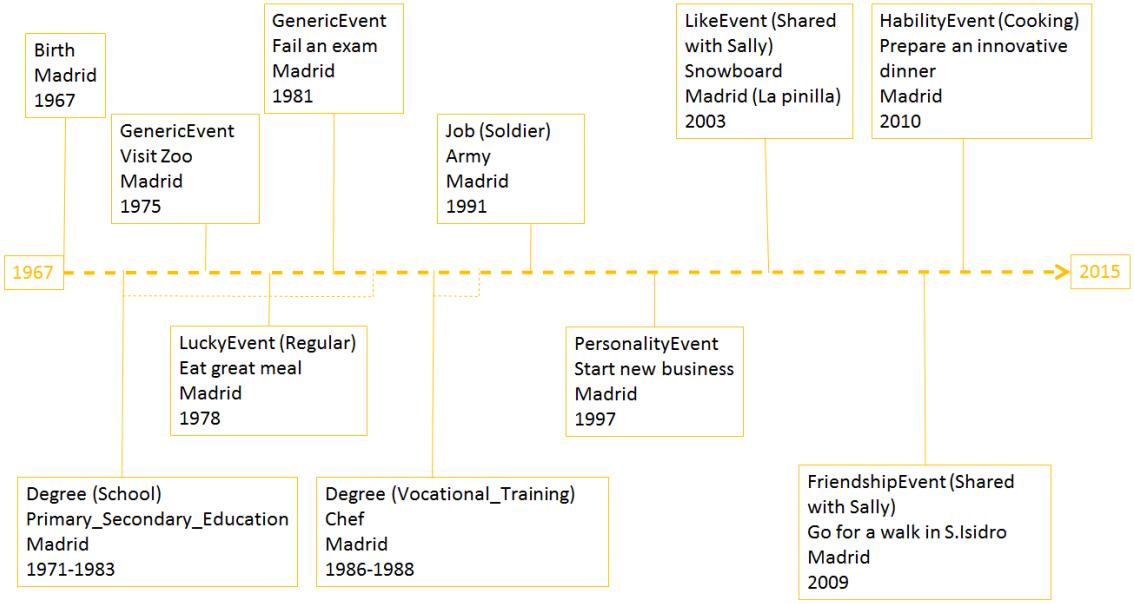

Ilustración 38: Historia Charlie. 2 ejecución

Se ha creado la historia de Charlie y vemos que ha vivido siempre en Madrid. Realizó estudios obligatorios y el grado medio de cocina al tener entre sus gustos la comida (Food).

Hay eventos al comienzo de tipo genéricos relativos a la edad que tenía, en donde fue al zoo y suspendió un examen y otro evento de tipo suerte en la que el personaje en sí tiene una suerte normal e indica que un día recuerda que comió una gran comida en su niñez.

Posteriormente se pone a trabajar como soldado, esto ha sido así puesto que para ser soldado debes tener estudios primarios y gustarte la acción (Action) cosa que Charlie cumpliría.

Encontramos un evento de tipo personalidad en donde se indica que Charlie comienza un nuevo negocio. En este caso, como en el de Sally, el que haya un margen de elevado para la personalidad hace que los eventos de este tipo no ofrezcan un punto objetivo relativo a la forma de ser que tiene.

Más tarde, encontramos un evento compartido con el anterior personaje, Sally, de tipo gustos. Este tipo de eventos sólo se ejecuta si los actores tienen un gusto en común. El gusto que tienen conjunto es el de montaña (Mountain) y es por eso que se van a hacer Snowboard a La Pinilla. Además, este evento ocurre a finales de noviembre y no en julio o agosto ya que se controla que este tipo de eventos deben cumplirse en determinadas épocas del año.

El presentar ya un evento compartido entre actores, hace que los eventos de amistad se puedan ejecutar. Esto es así debido a que todos los eventos de tipo amistad precisan que anteriormente hayan tenido eventos en común de otros tipos en señal de que estos sims se conocían anteriormente y es por eso que son amigos.

A continuación encontramos un evento de tipo amistad entre Sally y Charlie. Los eventos tipos de amistad hacen que los dos actores realicen la acción cuando sólo a uno de ellos le gusta realizarla pero el otro personaje lo acepta por considerarse amigos. El evento es pasear por San Isidro y se realiza porque a Sally le gusta pasear (Walking).

Para finalizar con la historia de Charlie, encontramos un evento de habilidad relacionado con la cocina puesto que la tiene como una de sus habilidades (Cooking).

En esta ejecución la historia de Sally (creada ya en la primera ejecución) ha interactuado con la de Charlie al compartir gustos y haber forzado la creación de eventos de amistad. La historia de Sally (gráficamente) aumentaría en dos nuevos eventos quedando de la siguiente forma:

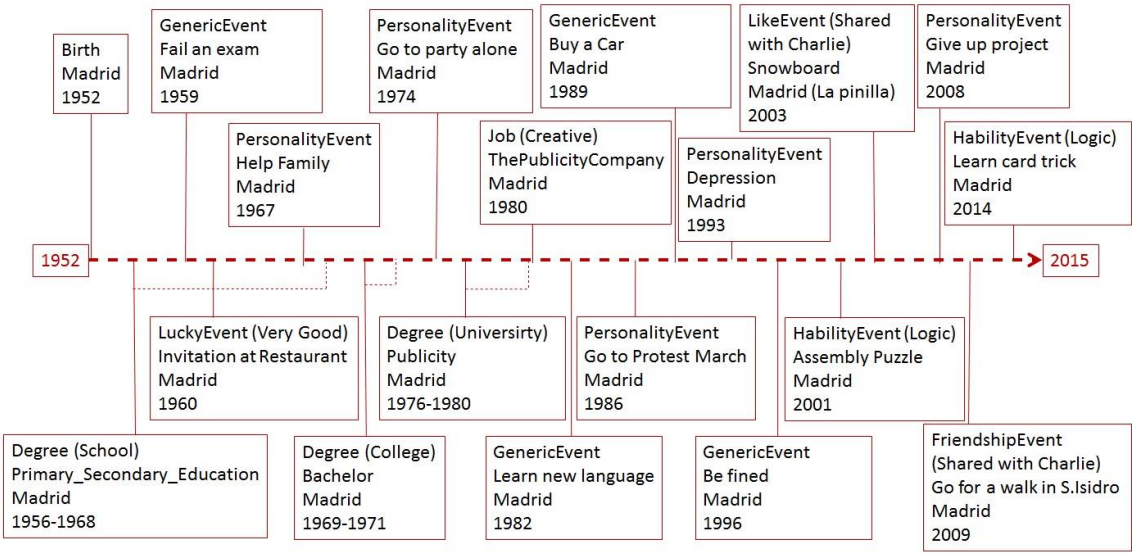

Ilustración 39: Historia Sally. 2 ejecución

Una vez ejecutado ambos personajes hemos podido verificar que, según van entrando personajes a AI-LIVE, las historias de los personajes ya presentes en el universo pueden aumentar gracias a las relaciones que se van creando entre sí y que, otorgando mayor peso a la creación de algunos eventos, se priorizan unos eventos frente a otros.

A continuación mostramos cómo quedan las historias de ambos historias al ejecutarse ambos personajes:

Sally:

---*STORY OF [CLIPS_ACTOR_HOGPO2LMQU]*---

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind Birth in Madrid at 24/1/1952.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind Degree in Madrid from 10/9/1956 until 4/6/1968. CLIPS_ACTOR_HOGPO2LMQU studied Primary_Secondary_Education at School.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind GenericEvent in Madrid at 28/1/1959. The event was fail an exam.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind LuckyEvent in Madrid at 30/1/1960. This was a Very Good luck event and it was get an invitation in a restaurant for being client number 1000.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind PersonalityEvent in Madrid at 30/10/1967. This event was help family with the familiar business.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind Degree in Madrid from 17/9/1969 until 19/6/1971. CLIPS_ACTOR_HOGPO2LMQU studied Bachelor at College.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind PersonalityEvent in Madrid at 14/9/1974. This event was go to a party where everyone is unknown people.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind Degree in Madrid from 6/9/1976 until 10/6/1980. CLIPS_ACTOR_HOGPO2LMQU studied Publicity at University.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind Job in Madrid at 28/11/1980. CLIPS_ACTOR_HOGPO2LMQU worked as a Creative at ThePublicityCompany.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind GenericEvent in Madrid at 3/8/1982. The event was learn a new language.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind PersonalityEvent in Madrid at 21/10/1986. This event was go to a protest march in order to fight for human rights.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind GenericEvent in Madrid at 2/4/1989. The event was buy a car.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind PersonalityEvent in Madrid at 18/9/1993. This event was be home depressed.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind GenericEvent in Madrid at 9/5/1996. The event was be fined.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind HabilityEvent in Madrid at 20/5/2001. The event's nature was Logic and the event itself was assembling a 500 pieces' jigsaw puzzle.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind LikeEvent in Madrid at 30/11/2003. The event was practice snowboarding in La pinilla mountain. This event occurred with CLIPS_ACTOR_HQERU4NRBH.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind PersonalityEvent in Madrid at 21/3/2008. This event was give up an important project.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind FriendshipEvent in Madrid at 27/8/2009. The event was accompany go for a walk in San Isidro. This event occurred with CLIPS_ACTOR_HQERU4NRBH.

Sim CLIPS_ACTOR_HOGPO2LMQU had an event of kind HabilityEvent in Madrid at 12/9/2014. The event's nature was Logic and the event itself was learn a new card trick.

---END OF STORY OF [CLIPS_ACTOR_HOGPO2LMQU]---

Charlie:

---STORY OF [CLIPS_ACTOR_HQERU4NRBH]---

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind Birth in Madrid at 6/10/1967.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind Degree in Madrid from 5/9/1971 until 1/6/1983. CLIPS_ACTOR_HQERU4NRBH studied Primary_Secondary_Education at School.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind GenericEvent in Madrid at 19/6/1975. The event was go to the zoo.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind LuckyEvent in Madrid at 26/8/1978. This was a Regular luck event and it was eat a great meal.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind GenericEvent in Madrid at 6/11/1981. The event was fail an exam.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind Degree in Madrid from 15/9/1986 until 14/6/1988. CLIPS_ACTOR_HQERU4NRBH studied Chef at Vocational_Training.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind Job in Madrid at 14/9/1991. CLIPS_ACTOR_HQERU4NRBH worked as a Soldier at Army.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind PersonalityEvent in Madrid at 3/9/1997. This event was start a new business.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind LikeEvent in Madrid at 30/11/2003. The event was practice snowboarding in La pinilla mountain. This event occurred with CLIPS_ACTOR_HOGPO2LMQU.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind FriendshipEvent in Madrid at 27/8/2009. The event was accompany go for a walk in San Isidro. This event occurred with CLIPS_ACTOR_HOGPO2LMQU.

Sim CLIPS_ACTOR_HQERU4NRBH had an event of kind HabilityEvent in Madrid at 16/11/2010. The event's nature was Cooking and the event itself was cooking an innovative dinner.

---END OF STORY OF [CLIPS_ACTOR_HQERU4NRBH]---

- **Prueba 3: Ejecución de juego en donde un sim aprende habilidad gracias a que otro sim le recuerda evento de habilidad pasado.**

Esta prueba sirve para verificar que un sim puede otorgar conocimiento a otro con habilidad inferior en un campo mediante el recuerdo del evento de esa habilidad.

Para poder comprobar que un sim aprende habilidad de otro gracias a un evento pasado de habilidad será necesario, lo primero de todo, que haya una historia con, al menos, un evento de habilidad correspondiente.

Se han ejecutado dos actores:

./run.sh -s -c amy -c charlie

Características de Amy:

```
([DEFAULT_ACTOR] of AddClient
(name_ "Amy")
(gender F)
(agreeableness 0.5)
(conscientousness 0.5)
(extraversion 0.5)
(neuroticism 0.5)
(openness 0.5)
(age 55)
(status 0)
(sex 1)
```

(theoretical 4)
(practical 2)
(modo Hability)
(habilities Logic Cooking)
(level_hability 7 2)
(generLike Kids Animals Dancing Art Mountain Walking History Beauty Pets)

Características de Charlie:

([DEFAULT_ACTOR] of AddClient
(name_ "Charlie")
(gender M)
(agreeableness 0.9)
(conscientousness 0.5)
(extraversion 0.6)
(neuroticism 0.4)
(openness 1)
(age 48)
(status 0)
(sex 0)
(theoretical 4)
(practical 2)
(modo Hability)
(habilities Cooking Sport Logic)
(level_hability 6 5 3)
(generLike Action Films BallSports Snow Mountain Food Pets Photo)

Al realizar la ejecución de los dos actores se genera la historia de cada uno de ellos. Lo que buscamos es que, al menos uno, tenga un evento de habilidad. A continuación mostramos cada una de las historias:

Amy:

---STORY OF [CLIPS_ACTOR_JBBNXEXG0H]---

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind Birth in Barcelona at 17/9/1952.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind Degree in Barcelona from 17/9/1956 until 4/6/1968. CLIPS_ACTOR_JBBNXEXG0H studied Primary_Secondary_Education at School.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind GenericEvent in Barcelona at 29/3/1961. The event was fail an exam.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind LuckyEvent in Barcelona at 8/4/1962. This was a Good luck event and it was go to Disneyland and meet Mickey Mouse.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind GenericEvent in Barcelona at 10/7/1964. The event was pass an exam.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind HabilityEvent in Barcelona at 2/1/1966. The event's nature was Logic and the event itself was solve a sudoku.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind Degree in Barcelona from 15/9/1972 until 1/6/1974. CLIPS_ACTOR_JBBNXEXG0H studied Bachelor at Bachelor.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind Degree in Barcelona from 8/9/1981 until 8/6/1983. CLIPS_ACTOR_JBBNXEXG0H studied Vet-assistant at Vocational_Training.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind PersonalityEvent in Barcelona at 11/4/1990. This event was donating blood in the hospital.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind LuckyEvent in Barcelona at 24/9/1995. This was a Good luck event and it was find a suitcase full of money.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind LikeEvent in Barcelona at 26/1/2002. The event was ride a snow motorbike in Montseny Natural Park.This event occurred with CLIPS_ACTOR_OX1HH9UOWY.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind GenericEvent in Barcelona at 19/8/2004. The event was go on strike.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind Move in Barcelona at 6/8/2005.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind PersonalityEvent in Barcelona at 16/10/2008. This event was go to a protest march in order to fight for human rights.

Sim CLIPS_ACTOR_JBBNXEXG0H had an event of kind LuckyEvent in Barcelona at 6/11/2012. This was a Good luck event and it was winning a tv contest.

---END OF STORY OF [CLIPS_ACTOR_JBBNXEXG0H]---

Charlie:

---STORY OF [CLIPS_ACTOR_OX1HH9UOWY]---

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind Birth in Barcelona at 29/8/1967.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind Degree in Barcelona from 18/9/1971 until 2/6/1983. CLIPS_ACTOR_OX1HH9UOWY studied Primary_Secondary_Education at School.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind GenericEvent in Barcelona at 2/1/1975. The event was go to a camp.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind Pet in Barcelona from 31/5/1978 until 9/10/1979. CLIPS_ACTOR_OX1HH9UOWY had a Turtle called Tobby.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind GenericEvent in Barcelona at 24/7/1980. The event was pass an exam.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind PersonalityEvent in Barcelona at 24/6/1981. This event was sponsor an indigenous child.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind HabilityEvent in Barcelona at 19/12/1982. The event's nature was Logic and the event itself was decipher hieroglyphic.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind Degree in Barcelona from 7/9/1983 until 13/6/1985. CLIPS_ACTOR_OX1HH9UOWY studied Chef at Vocational_Training.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind PersonalityEvent in Barcelona at 4/12/1986. This event was go to a party where everyone is unknown people.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind Degree in Barcelona from 18/9/1989 until 5/6/1993. CLIPS_ACTOR_OX1HH9UOWY studied Publicity at University.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind HabilityEvent in Barcelona at 3/12/1993. The event's nature was Cooking and the event itself was seasoning dessert by mistake.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind Job in Barcelona at 2/4/1997. CLIPS_ACTOR_OX1HH9UOWY worked as a Chef at Txistu.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind PersonalityEvent in Barcelona at 18/5/2000. This event was give up an important project.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind LikeEvent in Barcelona at 26/1/2002. The event was ride a snow motorbike in Montseny Natural Park.This event occurred with CLIPS_ACTOR_JBBNXEXG0H.

---END OF STORY OF [CLIPS_ACTOR_OX1HH9UOWY]---

Si leemos las historias, comprobamos que ambos actores tienen eventos de habilidad. Observando las habilidades en común encontramos la de Lógica (Logic) pero el actor *CLIPS_ACTOR_OX1HH9UOWY* (que corresponde a Charlie por la fecha de nacimiento y edad) será el único que pueda hacer uso de la regla "RememberStory" ya que es quien tiene mayor nivel en Lógica (3 frente a 2 de Amy) y Amy será quien se beneficie de esta habilidad aumentando su capacidad.

Sim CLIPS_ACTOR_OX1HH9UOWY had an event of kind HabilityEvent in Barcelona at 19/12/1982. The event's nature was Logic and the event itself was decipher hieroglyphic.

Así pues, vamos a ver como es el actor *CLIPS_ACTOR_OX1HH9UOWY* quien hace uso de esa regla y *CLIPS_ACTOR_JBBNXEXG0H* quien aumenta la habilidad:

*********** regla REMEMBERHABILITYSTORY **********

- El actor [CLIPS_ACTOR_OX1HH9UOWY] le recuerda una habilidad de Logic a: [CLIPS_ACTOR_JBBNXEXG0H]

([gen325] of Hability

(actor [CLIPS_ACTOR_JBBNXEXG0H])

(level 2)

(type Logic)

(available TRUE)

(complete FALSE)

(achieved 0)

(necesary 0)

(accumulated 2)

(hability_required Neither)

(tree FALSE)

(stateTree NEITHER)

)

De esta manera hemos podido comprobar que un actor con eventos pasados en su historia de habilidad puede enseñar a otro personaje con una misma habilidad en común pero con menor nivel que el primero, haciendo que el proyecto de generación automática de historias de personajes no solo se limita a generar eventos sino que se integra con el resto de funcionalidades de AI-LIVE y añade nuevas posibilidades al universo.

6. Gestión del Proyecto

A continuación se va a proceder a documentar cómo se ha gestionado el proyecto. Este apartado se subdividirá en varios para abarcar la parte de estimación de tiempos así como el coste económico de la realización del proyecto. Este coste se va a tener que dividir entre costes directos (personal, material necesario para el proyecto) y costes indirectos (Climatización, agua, luz…).

6.1 Estimación de tiempos

Para poder realizar una estimación de tiempos en el proyecto hay que tratar de normalizar y dividir el trabajo de forma que se pueda unificar las horas empleadas en el proyecto para poder plasmarlas en una línea temporal.

Así pues, el proyecto se ha desarrollado en distintas etapas:

- Consultoría inicial: Se puede definir como la toma de requisitos necesarios para poder comenzar con el proyecto. Se abordará, en primer lugar, a un alto nivel para ir bajando poco a poco la abstracción y tener claras las dudas que puedan plantearse.
- Planificación del sistema: Esta etapa consiste en la preparación de los sistemas para poder abarcar la solución. Esto significa que tendremos que obtener el hardware y el software necesario así como la capacitación previa en las tecnologías y lenguaje de programación a usar.
- Diseño del sistema: Es la fase en donde se determinan los aspectos del diseño, preparando las metodologías que van a ser usadas para el correcto funcionamiento de la solución.
- Implantación: Esta etapa es la que más tiempo va a consumir. Se trata del desarrollo en código de la solución que se ha diseñado en la etapa previa. Se tendrá que acoplar el nuevo proyecto a AI-LIVE sin impactar en las funcionalidades ya creadas anteriormente.
- Pruebas: Esta etapa engloba tanto el tiempo empleado en el diseño de las pruebas a ejecutar cómo el tiempo en estudiar los resultados de las mismas pruebas. Es de vital importancia emplear bastante tiempo en crear pruebas para confirmar que el proyecto está funcionando como se le espera.

- Documentación: Finalmente, como todo buen proyecto, es necesario realizar un esfuerzo para poder disponer de la documentación que permita explicar a fondo cómo funciona el proyecto diseñado.

Debido a motivos personales y el hecho de tener que compaginar el trabajo con la realización de este proyecto, el tiempo que se le ha dedicado al mismo ha sido muy elevado ya que había ocasiones en las que no podía trabajar en ello durante semanas y en otras ocasiones podía estar horas varios días seguidos. Teniendo esto en cuenta, y para intentar unificar el tiempo empleado en la ejecución del proyecto de forma que se pueda mostrar en un diagrama de Gantt, se van a suponer una media de horas empleadas con una fecha inicio y fin que resultarían válidas de haber sido posible el dedicarle todo el tiempo posible al mismo.

Es por ello que vamos a suponer una media de 3 horas diarias de lunes a viernes con fines de semana libres comenzando el 1 de julio de 2014 y un mes de vacaciones en agosto de 2014.

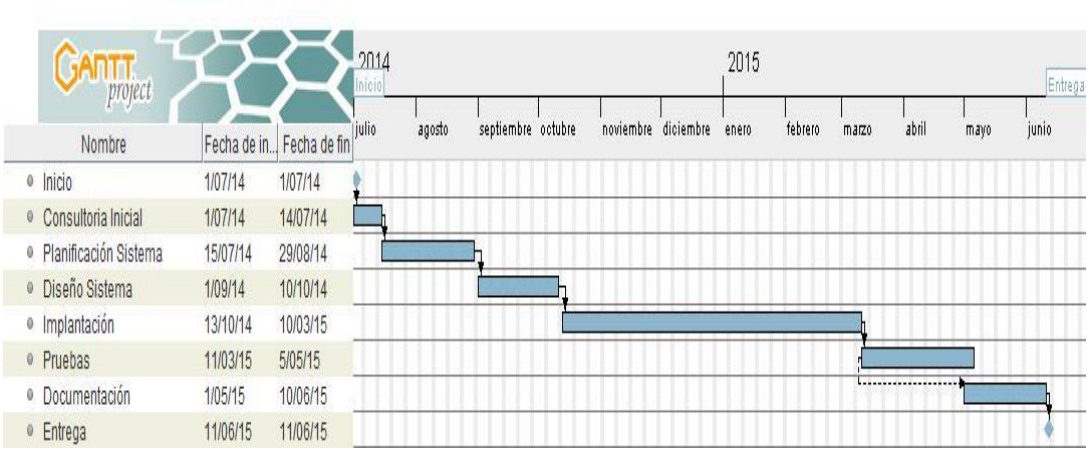

Ilustración 40: Diagrama Gantt

6.2 Presupuesto

Este apartado va a detallar los costes directos y los indirectos con el fin de poder cuantificar, económicamente hablando, lo que va a implicar llevar a cabo el proyecto.

6.2.1 Costes Directos

Los costes directos, al incluir los costes propios del personal, son los que representan el mayor porcentaje del total del presupuesto.

6.2.1.1 Costes Personal

Hay que tener en cuenta que para poder cuantificar el coste que nos supondría pagar al personal tenemos que obtener el coste/hora del programador a realizar el proyecto. Suponiendo un programador junior en España está cobrando alrededor de 15-20 euros/hora vamos a tomar la media (17,5€) y el resultado (según las fases usadas en el apartado anterior) sería el que aparece a continuación:

ETAPAS	HORAS	COSTE IMPORTE (€)
Consultoría Inicial	30	**525**
Planificación del Sistema	100	**1750**
Diseño del Sistema	90	**1575**
Implementación	320	**5600**
Pruebas	120	**2100**
Documentación	240	**4200**
TOTALES	**900**	**15750**

Tabla 1: Estimación de tiempos y costes de personal

6.2.2.2 Costes Materiales

Los costes materiales son los que entendemos como aquellos gastos necesarios, y tangibles, distintos a los personales que nos van a hacer falta para poder desarrollar el proyecto con seguridad.

Estos materiales serán usados durante todas las etapas del proyecto y para su coste se deberá tener en cuenta el plazo de amortización (en este proyecto se tiene un plazo de amortización de 3 años o 36 meses).

Los detallamos en la siguiente tabla:

ELEMENTO	DESCRIPCIÓN	PVP con IVA (€)	VALOR A AMORTIZAR (€)	MIN. MESES PARA AMORTIZAR	MESES	COSTE (€)
Hardware	Toshiba Satellite Pro R50 B 119	582,39	**481,31**	**36**	**12**	**160,44**
	Canon i-SENSYS MF8280Cw	304,18	**251,39**	**36**	**12**	**83,76**
Software	Ubuntu 10.4	0	**0**	**0**	**0**	**0**
	Otros software gratuitos	0	**0**	**0**	**0**	**0**
TOTALES		**886,57**	**732,7**			**244,20**

Tabla 2: Estimación costes materiales

En este caso, la dotación de amortización a aplicar es 244,20€.

6.2.2 Costes Indirectos

Los costes indirectos son aquellos que están asociados a gastos necesarios que se consumen durante el proyecto. De esta manera, podemos estar refiriéndonos a la luz, climatización o agua consumida.

Estos costes se consideran que son el 10% del coste directo por lo que la cifra en este caso es 1599,42€.

6.2.3 Resumen Presupuesto

La siguiente tabla muestra el resumen de todos los costes involucrados en el proyecto, así como la suma de los mismos:

TIPO	SUBTIPO	COSTE IMPORTE (€)
Coste Directo	Costes Personal	**15750,00**
	Costes Materiales	**244,20**
Costes Indirectos		**1599,42**
TOTALES		**17593,62**

Tabla 3: Estimación coste total

No se han tenido en cuenta el margen de riesgos de beneficios para este proyecto por normativa de la Universidad Carlos III de Madrid, centrándose exclusivamente en gastos generados para el desarrollo del mismo.

7. Conclusiones

Es el momento de exponer las conclusiones obtenidas tras la realización de este proyecto. En primer lugar, y desde el punto de vista del proyecto AI-LIVE, el proyecto ha añadido un punto más de realismo al gran proyecto que tantos alumnos años atrás han ido incorporando poco a poco.

Se ha creado una generación automática de historias que se basa en el sim que se va a crear y eso hace que su historia no desentone con su personalidad, sus habilidades o sus gustos. Estas historias van a acompañar al actor durante toda la vida en el juego AI-LIVE y va a permitir que sean explotados para posibles interacciones sociales. Para intentar mejorar las historias, se permite que una historia ya creada pueda ser ampliada con nuevos eventos que hayan ocurrido entre varios sims.

Se ha puesto especial hincapié en que la historia de cada personaje sea diferente al del resto. Se han implementado multitud de reglas variadas para crear distintos eventos que se han nutrido de instancias predefinidas aportando diferenciación con el resto de historias de personajes. Esto puede ofrecer miles de historias distintas en donde se podrán ver eventos que se repitan entre personajes pero será prácticamente imposible el que haya una historia con todos y cada uno de sus eventos idénticos a los de otra historia de otro personaje.

Desde el punto de vista del rendimiento, apenas se ha impactado negativamente en la aplicación AI-LIVE. Se hizo un estudio previo al desarrollo y se determinó que lo mejor sería implementar la solución al inicio de cada personaje en AI-LIVE en la parte del servidor. Cuando un personaje accede a AI-LIVE, es necesario que se genere la instancia “Actor” del personaje que tendrá sus características tales como personalidad, gustos por los objetos o habilidades y esto se generará una única vez al inicio al igual que su historia.

Se hicieron pruebas de medición antes y después de realizar la funcionalidad de Generación Automática de Historias y la diferencia era de 1 segundo más por personaje siendo este tiempo el que tardaba la aplicación en imprimir en la pantalla del servidor la historia del personaje al igual que imprime el resto de características del Actor.

Una vez se generaba el personaje, y se mostraba la pantalla del cliente, la comunicación entre cliente-servidor era la misma con o sin la nueva funcionalidad por lo

que demostraba que la decisión de implementar la solución en el servidor al inicio del juego había sido un acierto.

De esta manera, y en comparación a cómo estaba la aplicación sin la funcionalidad de Generación Automática de Historias, apenas se muestra latencia ni lentitud a la hora de iniciar el juego.

El hecho de tener que programar en CLIPS este proyecto ha sumado el grado de dificultad, ya que el diseño del sistema hacía necesario adecuar las distintas reglas o funciones disponibles para la creación de historias a la naturaleza propia de CLIPS con sus limitaciones. La lógica de CLIPS es ir ejecutando ciclos y, en cada ciclo, ir eligiendo la acción más idónea a realizar. Pero, para mi proyecto, lo interesante era que, una vez comenzada la creación de la historia del sim, TODOS los ciclos posteriores fueran propios de los eventos de la historia y no dejar elegir nada más hasta terminar la creación de la historia.

Otro problema encontrado a la hora de crear el proyecto ha sido la poca documentación relacionada con CLIPS y todo el potencial que puede ofrecer. Es lógico, ya que estamos hablando de un lenguaje embebido en C cuyo uso no está tan extendido como otros lenguajes tipo JAVA o el propio C.

Pese a la sencillez del lenguaje, en ocasiones se echaba en falta un compilador o depurador para comprobar errores o poder trazar cada una de las reglas que iban ejecutándose haciendo más lenta la parte de implantación y pruebas.

Aunque el proyecto en sí consiste en crear historias aleatorias se ha creado un par de acciones disponibles para los clientes que tienen relación con la historia con el fin de poder hacer uso de esta nueva funcionalidad.

Aprovechando la funcionalidad, que creó la compañera Pilar Blanco, de las habilidades de sim, se ha decidido darle una vuelta de tuerca para ofrecer un nuevo punto de vista para mejorar la habilidad: Que otro sim que tenga más habilidad y con un evento de historia de habilidad le explique ese evento a otro de forma que aprenda.

Los objetivos del proyecto se han visto cumplidos ya que, no solo se han creado historias automáticas sin perjudicar al resto de funcionalidades ya creadas, sino que se han creado historias con lógica sobre cada uno de los personajes haciendo que cada sim tenga su historia personal distinta a cualquier otro sim.

Para acabar, es interesante indicar todas las posibilidades que las historias van a poder ofrecer como nuevas ideas a desarrollar y ampliar la aplicación AI-LIVE.

8. Líneas futuras de trabajo

AI-LIVE es un macro proyecto alimentado de proyectos de alumnos de la Universidad Carlos III de Madrid. Poco a poco se van consiguiendo añadir funcionalidades para tener una aplicación robusta y muy versátil con la finalidad de crear un auténtico simulador social. Tras haber finalizado el proyecto final de carrera y haber testeado durante horas la aplicación habría varías líneas con las que seguir mejorando AI-LIVE.

8.1 Guardar progreso de juego y cargar partida

A día de hoy, AI-LIVE permite iniciar una partida haciendo uso de un perfil y el usuario va a poder estar haciendo uso de ese usuario hasta que decida dar por finalizada esa partida.

Una posible mejora sería incorporar la opción de poder guardar la partida antes de finalizarla de manera que se pueda volver a cargar el sim que estuve usando en un futuro para seguir mejorándolo o interactuando con otros actores.

8.2 Configurar el tipo de sim con el que jugar

Para poder acceder a AI-LIVE, en la consola de inicio se debe indicar el sim con el que iniciar la partida. Estos actores tienen preconfigurados todos los valores para poder comenzar la partida (edad, sexo, personalidad, habilidad o gustos).

Sería interesante que, al intentar acceder a AI-LIVE te permita seleccionar la opción de configurar manualmente el tipo de sim con el que entrar a jugar de manera que una persona pueda crear un actor que pueda ser una copia de sí mismo con sus gustos, edad, personalidad o habilidades.

Esto otorgaría un punto de mejora muy grande a la aplicación puesto que el usuario podría manejar su propia persona virtual y podría simular cómo se comportaría bajo situaciones que en la vida real no haría.

8.3 Mejorar la interfaz gráfica

Si se pretende conseguir una mejor y mayor jugabilidad en AI-LIVE es obligatorio mejorar la interfaz gráfica.

Es prácticamente inconcebible pensar en un juego bueno sin una buena interfaz en donde el usuario pueda tener una visión mucho más global del mundo AI-LIVE y que permitan realizar un análisis mayor de las necesidades de su agente.

8.4 Mejorar las uniones interpersonales

AI-LIVE cuenta con una parte relacionada con las relaciones interpersonales en las que se indican los valores que definen la amistad o cercanía que une a cada par de sims.

Además de esto, hay un atributo que hace referencia al tipo de unión que tienen estos sims. Puede ser una unión de sangre (familia) o amistad pero a día de hoy esta parte está muy poco desarrollada y sería realmente interesante poder darle una vuelta de forma que sea de utilidad.

Es posible que, ahora que ya hay una historia por cada actor, haciendo uso de los eventos en los que comparten historias entre sims se podría tomar una determinación sobre el tipo de unión que tienen ya que, por ejemplo, hay eventos de paternidad por lo que claramente eso podría indicar que el tipo de unión sería de sangre. También encontramos eventos de amistad que podrían indicar que tengan una unión de amistad más grande que otros.

8.5 Creación de nuevos escenarios

AI-LIVE tiene una serie de escenarios que, quizás, se empiecen a antojar algo escasos. No creo que sea necesario crear un mundo enorme con todos los objetos y acciones disponibles que tenemos en los escenarios actuales pero si sería una buena opción permitir nuevos escenarios con acciones distintas y posibilidades nuevas que hicieran mejorar la capacidad de juego de AI-LIVE.

Imaginémonos un escenario de tipo PLAYA. En ese escenario habrá nuevas acciones disponibles a realizar teniendo en cuenta las necesidades del actor y las consecuencias de abordarlas repercutirían igualmente en el sim.

8.6 Crear nuevas acciones basándose en las posibilidades actuales

Una vez cubiertas casi todas las acciones que implican satisfacer las necesidades básicas de un sim tales como comer, asearse, beber, comer o descansar, AI-LIVE está buscando mejorar la jugabilidad mediante otras acciones que tienen que ver con la vida cotidiana de una persona real como puede ser mejorar habilidades o mantener relaciones interpersonales.

Es por ello que sería interesante disponer de acciones que impliquen cambios en la persona en sí como pudiera ser realizar búsquedas de trabajo teniendo en cuenta las habilidades, gustos o personalidad del actor.

8.7 Crear nuevos eventos de historia con las acciones del juego AI-LIVE

A día de hoy, las historias que se generan para los personajes se basan en características propias de cada personaje y se van generando de forma automática para tener una serie de eventos lógicos.

Sería interesante que se pudieran ir añadiendo nuevos eventos a la historia de cada personaje con acciones cotidianas que puedan ir ocurriendo mientras se juega en el universo AI-LIVE como pudiera ser completar una habilidad al máximo.

9. Bibliografía

- Documentación de Proyectos de Fin de Carrera Previos
  - [PÉREZ, 2006] Proyecto de Fin de Carrera: AI-Live. Miguel Alfonso Pérez Bonomini. Universidad Carlos III de Madrid. 2006.
  - [BENITO, 2007] Proyecto de Fin de Carrera: Necesidades básicas de actores en universos de realidad simulada. Miguel Benito García. Universidad Carlos III de Madrid. 2007.
  - [JIMÉNEZ, 2008] Proyecto de Fin de Carrera: Diseño e implementación de un modelo comunicativo emocional para agentes virtuales en el universo AI-Live. Marta Jiménez Matarranz. Universidad Carlos III de Madrid. 2008.
  - [ESCUDERO, 2011] Proyecto de Fin de Carrera: Ampliación y mejora del universo virtual AI-Live. Javier Escudero Moreno. Universidad Carlos III de Madrid. 2011.
  - [UCEDA, 2012] Proyecto de Fin de Carrera: Desarrollo e implementación de un cliente para el entorno virtual de AI-Live. Alberto Uceda Blanco. Universidad Carlos III de Madrid. 2012.
  - [BLANCO, 2013] Proyecto de Fin de Carrera: Desarrollo de Habilidades y Capacidades del sim en AI-LIVE. Pilar Blanco Martínez. Universidad Carlos III de Madrid. 2013.
  - [BLANCO, 2014] Trabajo Fin de Grado: Generación Automática de Narrativas: Aventura Fantástica. Francisco Javier Blanco Álvarez. Universidad Carlos III de Madrid. 2014.

- Bibliografía ordenada alfabéticamente por el apellido del primer autor
  - D. Borrajo, N. Juristo, V. Martínez y J. Pazos, “Inteligencia Artificial. Métodos y Técnicas”, Centro de Estudios Universitarios Ramón Areces, Madrid, 1993.
  - Castle, L. 1998. “The Making of Blade Runner, Soup to Nuts!” In Proceedings of the Computer Game Developers' Conference, Long Beach, CA, 87-97.
  - Castronova, E. “Synthetic Worlds: The Business and Culture of Online Games”, University of Chicago Press, 2005.

- Laird, J. E. 2000. It Knows What You're Going To Do: Adding Anticipation to a Quakebot. In Papers from the AAAI 2000 Spring Symposium on Artificial Intelligence and Interactive Entertainment,Technical Report SS-00-02, 41-50. AAAI Press.

- Webs ordenadas por el nombre de la web
  - AVANET. Roles para gestión de proyectos. Visitado en Junio de 2015: http://www.avanet.org/estableciendo-mejoras-y-definiendo-roles-en-el-desarrollo-de-software.aspx
  - BATANGA. Evolución de la Inteligencia Artificial de los videojuegos. Visitado en Mayo de 2015: http://gamer.batanga.com/4846/la-evolucion-de-la-inteligencia-artificial-en-los-videojuegos
  - CYTA. Métodos de Inteligencia Artificial. Visitado en Junio de 2015: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/inteligencia_artificial.pdf
  - ELOTROLADO. Géneros de videojuegos. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos
  - FIBUPC. Historia de los videojuegos. Visitado en Abril de 2015: http://www.fib.upc.edu/retro-informatica/historia/videojocs.html
  - GENERATION5. Cómo empezar con la Inteligencia Artificial. Visitado en Mayo de 2015: http://www.generation5.org/content/2004/howto02es.asp
  - MONOGRAFIAS. Redes Neuronales. Visitado en Junio de 2015: http://www.monografias.com/trabajos12/redneur/redneur.shtml
  - MONOSGRAFIAS. Inteligencia Artificial. Visitado en Julio de 2015: http://www.monografias.com/trabajos16/la-inteligencia-artificial/la-inteligencia-artificial.shtml
  - MUYINTERESANTE. El primer superéxito de los videojuegos. Visitado en Abril de 2015: http://www.muyinteresante.es/tecnologia/preguntas-respuestas/icual-fue-el-primer-superexito-en-videojuegos
  - RAZONARTIFICIAL. Evolución de la Inteligencia Artificial. Visitado en Mayo de 2015: http://razonartificial.com/inteligencia-artificial/

- ROCHESTER. Inteligencia Artificial en los juegos. Visitado en Mayo de 2015: http://www.cs.rochester.edu/~brown/242/assts/termprojs/games.pdf
- SCENEBETA. El origen de los videojuegos. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
- WIKIPEDIA. Inteligencia Artificial de los videojuegos. Visitado en Mayo de 2015: http://es.wikipedia.org/wiki/Inteligencia_artificial_%28videojuegos%29
- WIKIPEDIA. Primer videojuego. Visitado en Abril de 2015: http://es.wikipedia.org/wiki/Primer_videojuego

- Fotos ordenadas por número de foto dentro del documento:
  - 1: MISILES 1947. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
  - 2: OXO 1952. Visitado en Abril de 2015: http://en.wikipedia.org/wiki/OXO
  - 3: EDSAC 1952. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
  - 4: TENNIS FOR TWO. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
  - 5: OSCILOSCOPIO. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
  - 6: SPACE WARS. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
  - 7: PDP-1. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong

- 8: GALAXY GAME. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
- 9: COMPUTER SPACE. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
- 10: MÁQUINA COMPUTER SPACE. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
- 11: PONG 1972. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
- 12: PONG ATARI. Visitado en Abril de 2015: http://www.scenebeta.com/tutorial/1952-1972-el-origen-de-los-videojuegos-desde-oxo-hasta-pong
- 13: JET SET RADIO. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 14: IKARUGA. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 15: HALO 2. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 16: CAPTAIN COMMANDO. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 17: SUPER STREET FIGHTER 2. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 18: BLADE. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 19: CYBERIA. Visitado en Mayo de 2015: www.gfxtorrent.net

- 20: EL QUIJOTE. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 21: MONKEY ISLAND. Visitado en Mayo de 2015: www.meristation.com
- 22: BLACK & WHITE. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 23: VIVA PIÑATA. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 24: LOS SIMS. Visitado en Mayo de 2015: www.meristation.com
- 25: FABLE. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 26: SHINING FORCE. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 27: WOW. Visitado en Mayo de 2015: www.meristation.com
- 28: AGE OF EMPIRES 3. Visitado en Mayo de 2015: http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos#Acci.C3.B3n
- 29: KUNG FUU. Visitado en Mayo de 2015: http://www.craveonline.com/gaming/articles/145974-10-old-school-games-that-need-remakes
- 30: LOS SIMS. Visitado en Mayo de 2015: http://juegosabiertos.com/lossims/informacion.php

10. Anexos

En este punto se recogen los documentos que se han creado para tener una buena explicación sobre cómo funciona el juego AI-Live. Estos documentos están en continua actualización al tener que añadir cada alumno que incluye su proyecto nueva funcionalidad. Se ha decidido no incluir dentro de esta memoria cada documento al encontrarse en el repositorio que se utiliza para albergar el juego. Ambos documentos se encuentran tanto en versión pdf, como en versión .doc para que puedan ser editados en futuras versiones:

- Manual de Usuario: Documento que explica las premisas e indicaciones que se ha de seguir para instalar, compilar y ejecutar los diferentes módulos del juego. El manual se encuentra ubicado en el siguiente enlace: http://pruebaailivesvn.googlecode.com/files/Manual_de_usuario.pdf

- Manual de Referencia: Documento que trata de ayudar a entender mejor el juego AI-Live ya que en él se explica con detalle todas las reglas, métodos, funciones e instancias de las que se encuentra formado el juego AI-Live. Se encuentra ubicado en el siguiente enlace: http://pruebaailivesvn.googlecode.com/files/Manual_de_referencia.pdf

- Repositorio de AI-Live: Documento que trata de ayudar a manejar la herramienta de Google Code encargada de albergar el juego AI-Live. El documento explica cómo poder realizar acciones sobre el juego AI-Live, cómo subir modificaciones o bajar las diferentes versiones para los tres sistemas operativos más populares, Windows, Linux y Mac. Se encuentra ubicado en el siguiente enlace: http://pruebaailivesvn.googlecode.com/files/Repositorio_de_AI_LIVE.pdf

- Ontología: Documento que alberga de una forma gráfica la ontología que utiliza el juego AI-Live. Se encuentra ubicado en el siguiente enlace: http://pruebaailivesvn.googlecode.com/files/Ontologia.pdf